Technical University of Berlin
Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science
Algorithmics and Computational Complexity (AKT)

Towards Linear-Time Parameterized Algorithms for Strongly Stable Matching

Rosa Wolf

Thesis submitted in fulfillment of the requirements for the degree
Bachelor of Science (B. Sc.) in the field of Computer Science

March 2021

Supervisor and first reviewer: Prof. Dr. Rolf Niedermeier
Second reviewer: Prof. Dr. Markus Brill
Co-Supervisors: Klaus Heeger and Dr. André Nichterlein

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Neuenkirchen bei Greifswald, _____ _____

Date Signature

Zusammenfassung

Wir führen eine parametrisierte Komplexitätsstudie des in Polynomialzeit lösbaren Strongly Stable Matching Problems durch und präsentieren damit weitere Forschung auf dem relativ neuen Gebiet des FPT in P. Eine Instanz von Strongly Stable Matching ist ein ungerichteter Graph $G = (V, E)$ und eine schwach geordnete Präferenzliste für jeden Knoten in V. Ein Matching M ist *strongly stable*, wenn kein Knoten $v \in V$ einen Nachbarn $u \in V$ seinem Matchingpartner vorzieht, während Knoten u Knoten v mindestens genau so gerne wie seinen eigenen Matchingpartner mag.

Dieses Problem kann in $\mathcal{O}(|V| \cdot |E|)$ Zeit gelöst werden. Obwohl dies eine recht akzeptable Laufzeit zu sein scheint, kann sie bei großen Netzwerken dennoch deutlich zu hoch sein. Mit dem Ziel den Anwendungsbereich zu vergrößern, stellen wir effiziente Datenreduktionsregeln vor, welche Knoten vom Grad null und Grad eins löschen sowie alle Pfade aus Knoten vom Grad zwei auf eine konstante Länge reduzieren. Die Datenreduktionsregeln führen uns zum Hauptergebnis dieser Arbeit, einem lineargroßen, in Linearzeit berechenbaren Kernel für Strongly Stable Matching, parametrisiert durch die feedback edge number.

Abstract

We perform a parameterized complexity study of the polynomial-time solvable Strongly Stable Matching problem and thus present further research in the relatively new field of FPT in P. An instance of Strongly Stable Matching is an undirected graph $G = (V, E)$ and a weakly ordered preference list for every node in V. A matching M is *strongly stable* if no node $v \in V$ prefers a neighbor $u \in V$ to its matching partner while u likes v at least as much as its own matching partner.

This problem can be solved in $\mathcal{O}(|V| \cdot |E|)$ time. While this might appear to be a fairly acceptable running time, it can still become infeasible for large networks. Aiming to expand the applicable range, we present efficient preprocessing steps to delete degree-zero and degree-one nodes and reduce all paths of degree-two nodes to a constant length. The data reduction rules lead us to the main result of this thesis, a linear-size, linear-time computable kernel parameterized by the feedback edge number for Strongly Stable Matching.

Contents

Chapter 1

Introduction

In our society many matching problems and tasks are part of everyday life. As an example, we address the "zentrale Vergabeverfahren" being used in Germany to assign medical college applicants to the limited number of medical college places [Grü11]. In 2017 there were already more than 43000 applicants for medicine in Germany and, though significantly less, more than 9000 college places [Ste]. Thus, it is not hard to imagine that manually assigning students to college places is infeasible. Hence, we have to model the task as an algorithmic problem. The corresponding stable matching problems are the focus of this thesis.

Our goal is to assign the group of applicants to the group of colleges. In other words, we want to *match* the applicants with the college places. Thus, we are presented with a matching problem. However, we cannot just match any person to any college, but have to consider more aspects when assigning applicants to colleges. It would not make much sense to randomly admit the applicants to colleges. Applicants might be unhappy with certain colleges or even unwilling to enroll themselves. It would make much more sense to offer college places to applicants who then also accept the offer. Thus, we have to consider the preferences of each applicant concerning the different colleges. Moreover, we also have to draw into consideration the preferences of the colleges when choosing their students. Since most colleges have a limited number of places, it might be sensible to offer those places to students that seem to be more likely to finish their studies.

Thus, we have a preference list for every applicant and every college in our model. Our goal is to match the applicants to colleges such that each applicant is assigned to a college to their liking while also considering the preferences of the colleges for the applicants. If, for example, we have two applicants who both like a college the same, then we would offer the place to the applicant who is ranked higher in the college's preference list.

This results in a stable matching problem. Generally, in stable matchings each node can only be assigned to a maximum of one other node in the graph. Moreover, the graph is not partitioned. However, in our example we have two partitions, the applicants and the colleges. More specifically, we have one group where every member can only be assigned to one node of the other group, and one group where every node can be assigned to several members of the other group. I.e. the colleges can be assigned to several applicants and thus have a capacity. Hence, we are presented with the Hospital/Residents problem

which is also called College Admission problem. This problem is a kind of stable matching problem, where the input graph is bipartite and the input instance additionally has capacities for each node in one of the partitions [GS62].

Above we talked about the case in which two applicants like a college the same, but the college has distinct preferences between the applicants. However, we still do not know what to do in the case of two applicants who both prefer a college to the other colleges when the college also likes both applicants the same. Of course, if the college still has several open places, then both applicants could be admitted. But what do we do if only one college place is left? This is where the notion of stability becomes important. In Stable Matching our goal is, as the name already suggests, to find a matching that satisfies certain criteria of stability. The stability of a matching can be defined via *blocking* edges. If a blocking edge exists in a matching, then the matching is not stable. There exist different definitions of blocking edges. In this thesis we focus on one definition which is used for Strongly Stable Matching. Here, an edge $\{u, v\}$ is blocking, if:

- node u prefers being matched to node v to its current situation, i.e., to their current matching partner or to not being matched at all, and
- node v either prefers being matched to u to its current situation, or likes u the same as its current situation

Such blocking pairs $\{u, v\}$ have to be avoided, as they disregard the existing preferences. Applied to our example this specifically means that if we have a college which ranks two applicants the same and those two applicants cannot be matched to any other college which they like at least as much as this college, then we have a blocking edge. Because, if we admit one of the two applicants to the college, then the edge between the other applicant and the college is blocking. Thus, in this case a strongly stable matching does not exist. An example for the College Admission problem using strong stability is also visualized in Figure 1.1.

Above we presented the example of the college admission system. Stable Matching in centralized college admission systems is used in many countries, like China [Zha10] and the US [CKK19]. However, Stable Matching is used in many more applications than just college admission. Many applications that have to implement assignment problems use Stable Matching. For example in Japan a matching system was used to regulate the number of medical residents working in each prefecture of Japan [KK12]. Stable Matching can also be used in the assignment of trainees to software projects [Gha+15]. Moreover, organ-transplants often require complicated assignment systems. Huang [Hua10] suggested a Stable Matching-model to organize kidney-transplants.

It is obvious that there are many more potential applications of matching problems. In this thesis, we focus on Strongly Stable Matching which is a fundamental matching problem. Our central result is an efficient and effective preprocessing for Strongly Stable Matching.

1.1 Our Contributions

In the thesis, we will develop a linear-time computable, linear-size kernel for Strongly Stable Matching. Thus, we will fomulate linear-time computable data reduction

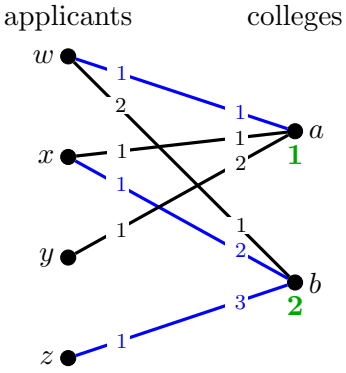


Figure 1.1: An example showing the College Admission problem. On the left we have the set of applicants and on the right the set of colleges. The number on the edge $\{u, v\}$, which is closer to u indicates the position of v in the preference list of u. The green numbers indicate the capacity of a college. The blue edges show the edges contained in the strongly stable matching, which we will discuss subsequently. College a has a capacity of 1. One of its first preferences is applicant w, which has college a as single first preference. Thus, edge $\{w, a\}$ must be in any stable matching. However, college a also has applicant x as first preference and x has w as first preference. For the edge $\{x, a\}$ not to be blocking, applicant x must be matched to another college it likes at least as much as college a. Hence, applicant x must be matched to its other first preference, college b. College b has a capacity of 2. Its only first preference is applicant w, which does not prefer b to its matching partner a. Thus, college b cannot be matched to w. College b's second preference is applicant x. Above we already concluded that edge $\{x, b\}$ has to be in matching. We can now conclude that this does not produce further blocking edges and thus is indeed possible. College b has one capacity left. Next in its preference list is applicant z, which has no neighbor besides b. Thus, edge $\{z, b\}$ is in the matching.

rules to enforce a specific graph structure, such that the size of the reduced graph will be linearly dependent on the feedback edge number. The feedback edge number of a graph is the number of edges that need to be deleted to obtain a tree or forest. Thus, it indicates how different the structure of the graph is of that of a tree or forest. To be able to do so, the data reduction rules will delete all degree-zero and degree-one nodes. Furthermore, they will reduce all subgraphs where all inner nodes have degree two, to a constant length. However, the data reduction rules do not directly work on instances of Strongly Stable Matching, but on an more expressive annotated version of Strongly Stable Matching. Thus, we will also present a linear-time computable reduction from Strongly Stable Matching to the annotated version of Strongly Stable Matching and vice versa.

1.2 Related Work

Strongly Stable Matching is just one of many matching problems. We can categorize matching problems by the kind of input they get, as well as the output we want to achieve. As already mentioned above, an instance of Strongly Stable Matching consists of a graph and preference lists for all nodes. The goal is to find a matching that satisfies all criteria of strong stability. However, there are other matching problems,

stable matching problem	running time and complexity	
	bipartite graphs	general graphs
without ties	$\mathcal{O}(n+m)$ [GS62]	$\mathcal{O}(n+m)$ [Irv85]
with ties		
weak stability	$\mathcal{O}(n+m)$ [GS62]	NP-complete [Ron90]
strong stability	$\mathcal{O}(nm)$ [Kav+04]	$\mathcal{O}(nm)$ [Kun16]
super stability	$\mathcal{O}(n+m)$ [Irv94]	$\mathcal{O}(n+m)$ [IM02]

Table 1.1: An overview over different problems of stable matching with preferences. We can categorize the problems into the group of problems in which ties are not allowed and the group in which ties are allowed. In the latter group we can further distinguish between the three different notions of stability.

where we do not have preference lists.

Classical Matching problems. The probably most basic variant of Matching problem is the Maximum-Cardinality Matching. The Maximum-Cardinality Matching gets a graph as input and returns a matching with the maximum number of edges that is possible in the graph. Hopcroft and Karp [HK73] proved that this problem can be solved in $\mathcal{O}(\sqrt{n}m)$ time in bipartite graphs, where n is the number of nodes and m the number of edges of the input graph. Their algorithm is based on finding augmenting paths. They showed that a maximal set of augmenting paths can be found in $\mathcal{O}(n+m)$ time, and that a maximum of $\mathcal{O}(\sqrt{n})$ phases is needed to find a maximum-cardinality matching in a bipartite graph. On the basis of Hopcroft and Karp's ideas an algorithm for finding a maximum-cardinality matching in general graphs in $\mathcal{O}(\sqrt{n}m)$ time was developed [MV80].

Another matching problems where the input does not contain preference lists is the Maximum-Weight Matching. Here the input is a graph and weights for every edge in the graph. The goal is to find a matching that maximizes the sum of the weights of the edges in the matching. Edmonds [Edm65] proved that a maximum weighted matching can be found in $\mathcal{O}(n^2m)$ time. If all weights are integers then the running time can be reduced to $\mathcal{O}(Nm\sqrt{n})$, where N is the maximum weight [Gab85].

Matchings with Preferences. We can further categorizes the matching problems having preferences in the input [Man13]. In all different cases our goal is to find a stable matching. The differences lie in the definition of stability. An overview over the different stable matching problems with preferences is also given in Table 1.1. We distinguish whether ties are allowed in the preference lists. If ties are not allowed, then a stable matching can be found in linear time [Irv85]. If ties are allowed, i.e., the preference lists are weakly ordered, we can distinguish between different stability notions. Above we already introduced blocking edges. Recall that we described the cases in which edges are blocking in Strongly Stable Matching. Besides strong stability there exist two further stability notions. In the Weakly Stable Matching problem, edges are blocking if there exist two nodes that are not matched to each other and strictly prefer

Figure 1.2: An example showing the three definitions of stability. The number on the edge $\{u, v\}$, which is closer to u indicates the position of v in the preference list of u. The matching M_1 which contains only the red edge is already weakly stable, as the nodes b and c are indifferent between both their partners. But M_1 is not strongly stable, as nodes a and d prefer their respective neighbor b or c to not being matched at all. Matching M_2 which contains the blue edges is strongly stable and thus weakly stable, but not super stable as the edge $\{b, c\} \notin M_2$ is not contained in the matching, but b and c are indifferent between both their neighbors. In the above graph no super stable matching exists.

each other to their partner in the matching. In Super-Stable Matching, edges are already blocking if two nodes that are not matched to each other are indifferent between each other and their partner in the matching. The different Stable Matching problems are also visualized in Figure 1.2. The Weakly Stable Matching problem was proven to be NP-complete [Ron90]. Irving [Irv94] showed that in bipartite graphs a super-stable matching can be found in linear time. Later it was shown that Super Stable Matching can be solved in linear time in any graph [IM02]. Strongly Stable Matching, on which we will concentrate in this thesis, can be solved in $\mathcal{O}(nm)$ time on general graphs [Kun16].

There exist different well-known special cases of stable matching problems. One example is the Stable Marriage problem. Stable Marriage is the special case of Stable Matching with preferences without ties where the input graph is a complete bipartite graph, i.e., a bipartite graph where every node of one partition has a common incident edge with every node of the other partition. The Stable Marriage problem can be solved by the famous Gale-Shapley algorithm in $\mathcal{O}(n^2)$ time [GS62].

Kunysz [Kun18] studied the special case of the Maximum Weighted Strongly Stable Marriage with incomplete lists and ties. By formulating a polyhedral characterization of the problem he shows that it can be solved in polynomial time. In contrast, Weighted Stable Matching on general graphs and without ties is NP-hard [Fed92].

Cseh and Juhos [CJ21] also researched Stable Marriage. They considered different levels of relaxations of preference lists in combination with all notions of stability. They analyzed for the different problems whether they can be solved in polynomial time or are NP-hard.

Matchings and Parameterized Complexity. In this thesis, we focus on parameterized complexity studies. There already exists much research on parameterized complexity of stable matching problems, especially for the NP-hard problem Weakly Stable Matching. Thus, all of the below mentioned important examples for parameterizations of Stable Matchings refer to the Weakly Stable Matching problem.

Marx and Schlotter [MS10] investigated the Stable Marriage problem with ties and incomplete lists. They showed that in this setting the NP-hard version of finding a maximum weakly stable matching is W[1]-hard with regard to the parameter number of ties. Moreover, the problem is fixed-parameter tractable with regard to the overall length of ties and para-NP-hard regarding the maximum length of ties [MS10].

Gupta, Saurabh, and Zehavi [GSZ17] researched further optimization versions of the

Stable Marriage problem, like max Stable Marriage with ties, or min Stable Marriage with ties. They showed that these problems are W[1]-hard, with regard to the parameter treewidth. Furthermore, they studied the parameter treewidth on different kind of input graphs and established the behaviour of the optimization versions of Stable Marriage on these graphs.

Adil et al. [Adi+18] investigated the Stable Marriage and Stable Roommate problem with ties and incomplete lists. They examined the optimization version of these problems, more specifically the maximization and minimization problem, which are NP-hard [Ron90]. For Stable Matching Adil et al. [Adi+18] showed that it is fixed-parameter tractable with respect to the solution size. For the Stable Roommate problem they showed that it is fixed-parameter tractable using the structural parameter vertex cover.

It was also proved that the Maximum-Cardinality Stable Roommate is W[1]-hard with regard to the parameters treedepth, tree-cut width and feedback vertex set [Bre+19]. Bredereck et al. [Bre+19] also proved that Perfect Stable Roommate and Stable Matching as a decision problem are fixed-parameter tractable with the parameters tree-cut width and feedback edge number.

Further research was done concerning special cases of the Stable Matching problem. Meeks and Rastegari [MR20] investigated Stable Matching where the input instance consists of k partitions that determine the preference lists of the agents in them. They showed that in this form the NP-hard problem Maximum Stable Marriage with ties and incomplete lists is fixed-parameter tractable regarding the number of types of agents.

The main research in the field of parameterized complexity of stable matching problems is done for NP-hard matching problems. We however want to concentrate on the Strongly Stable Matching, which is not NP-hard. While it might be obvious why the parameterized complexity of NP-hard problems is examined one could think that it is unnecessary to examine the parameterized complexity of a problem that can already be solved in cubic time. However, in practice often immensely large networks have to be handled. An example is the chinese college admission system that already had 10 million applicants in the year 2007 [Zha10]. Thus, even a running time of $\mathcal{O}(nm)$ can be unfeasible. Hence, every fast performing preprossessing step decreasing the graph size is beneficial. Thus, in the spirit of Mertzios, Nichterlein, and Niedermeier [MNN20] we will develop a linear-time computable kernelization for Strongly Stable Matching parameterized by feedback edge number. To this end, we will develop linear-time computable data reduction rules. The implementation of these data reduction rules is beyond the scope of this thesis. However, we point out that similar data reduction rules for Maximum Matching proved to be practically successful [Kor+18].

1.3 Outline

First we will introduce all necessary preliminaries used in this thesis. In Chapter 3 we will get acquainted with the topic of Strongly Stable Matching, by developing a linear-time algorithm for finding a strongly stable matchings in trees. In Chapter 4 we develop data reduction rules, show their correctness and prove their linear running time. However, these data reduction rules do not directly work on instances of Strongly Stable

Matching but an annotation of this problem. The reason for introducing this annotation is too extensive to explain here, but will be provided in the beginning of Chapter 4. The data reduction rules yield a linear-time computable, linear-size kernel for the annotated version of Strongly Stable Matching, which will be proven in Chapter 5. Here we will also provide polynomial-time many to one reductions between Strongly Stable Matching and the annotated version of Strongly Stable Matching. We use them in our main theorem: a linear-time computable, linear-size kernel for Strongly Stable Matching, which will be given at the end of Chapter 5.

Chapter 2

Preliminaries

In this chapter we will introduce notations and concepts which we will need in the course of this thesis. We use standard notations for graph theory.

Graph theory. Let $G = (V, E)$ denote an undirected graph, where V denotes the set of nodes and $E \subseteq \{\{v, w\} \mid v, w \in V, v \neq w\}$ denotes the set of edges.

We denote by

$V(G)$ the *node set* of G;

$E(G)$ the *edge set* of G with $E(G) \subseteq \binom{V(G)}{2}$; for an edge $e = \{u, v\} \in E(G)$ the two nodes u and v are called *endpoints* of e;

n_G the number $|V(G)|$ of *nodes*;

m_G the number $|E(G)|$ of *edges*;

$N_G(v)$ the (open) *neighborhood* of v, formally, $N_G(v) := \{u \in V \mid \{u, v\} \in E(G)\}$;

$\deg_G(v)$ the *degree* of v, formally, $\deg_G(v) := |N_G(v)|$.

If the referenced graph is clear from context we drop the subscript G. In this case we can also write V instead of $V(G)$ and E instead of $E(G)$.

Trees. A tree is an acyclic graph $G = (V, E)$. In this thesis we further specify that a tree is undirected. We fix an arbitrary node $r \in V$ as the root of the tree.

We denote by

$\text{depth}_G(v)$ the *depth* of v, which is equal to the number of edges of the shortest path from the root r to v;

$\text{height}(G)$ the *height* of G, which is equal to the highest depth among the depths of all leaves in G;

$\text{level}_G(i)$ the i^{th} *level* of G containing all nodes $v \in V$ with $\text{depth}_G(v) = i - 1$, formally $\text{level}_G(i) := \{v \in V \mid \text{depth}_G(v) = i - 1\}$;

$C_G(v)$ the *children* of $v \in V$, where G is a tree, formally $C_G(v) := \{u \in V(G) \mid \{v,u\} \in E(G) \wedge \text{depth}_G(u) = \text{depth}_G(v) + 1\}$.

A forest is a graph, where each connected component is a tree. If we have a forest instead of a tree the above definitions still hold for each connected component individually.

Stable Matching. We will now introduce stable-matching problems and related concepts.

There exist different kinds of stable matching problems depending on the definition of stability. The stable matching problems we will introduce here have in common that the input instance consists of an undirected graph $G = (V, E)$ and a preference list P_v for each node $v \in V$.

We will now take a more detailed look at preference lists. A preference list P_v of a node v is a tuple, that *weakly orders* a subset $N(v) \subseteq V(G)$. A weakly ordered set is a ordered set, where ties are allowed. Applied to preference lists this means that a node $v \in V$ can like several of its neighbors $u, w \in N(v)$ the same. In this case we also say that v is *indifferent* between u and w. We denote by $u \preceq_v w$ that v likes u at least as much as w. Thus, if v is indifferent between u and w, then it applies that $u \preceq_v w$ and $w \preceq_v u$.

If a node $u \in N(v)$ strictly precedes $w \in N(v)$ in the preference list of v, then we say that v *prefers* u to w. Here we also write $u \prec_v w$.

Let $|P_v| \leq n$ be the cardinality of the preference list P_v of v. Then we can also write $P_v = (Q_1, Q_2, ..., Q_{|P_v|})$, where $Q_1 \subseteq N(v)$ is the set of first preferences of v and $Q_{|P_v|} \subseteq N(v)$ is the set of last preferences of v. We denote by $\text{last}(v) := |P_v|$ the position of the last element in the preference list of v, i.e., the nodes, node v likes least. We denote by

$\pi_{i,...,j}(P_v)$ with $1 \leq i \leq j \leq |P_v|$ the projection of P_v to the weakly ordered set $P'_v = (Q_i, Q_{i+1}, ..., Q_j)$.

Furthermore, we define the binary operation $\cdot \cup \cdot$ in the special case of two one-element tuples $T_1 := (X)$ and $T_2 := (Y)$ with the sets X and Y as only elements, as follows

$$T_1 \cup T_2 = (X) \cup (Y) = (X \cup Y),$$

where $(X \cup Y)$ is another one-element tuple. We define the concatenation of two tuples of arbitrary length as follows

$$(X_1, X_2, ..., X_n) \circ (Y_1, Y_2, ..., Y_m) = (X_1, X_2, ..., X_n, Y_1, Y_2, ..., Y_m).$$

To be able to define specific matching problems, we first have to define what a matching is. Given a graph $G = (V, E)$ a matching $M \subseteq E$ is a subset of edges in G. Apart from some matching problems, matchings only consist of edges that are pairwise not incident to one another. For the matching problems we will introduce subsequently this also applies. Different to other matching problems in STABLE MATCHING we have to consider preferences. This leads us to the following definitions. In STABLE MATCHING our goal is to find a subset of edges $M \subseteq E(G)$ in the graph, such that no edge $\{u, v\} \notin M$ is *blocking*. There are three definitions of stability, which influence whether an edge is blocking. Let $\{u, v'\} \in M$ and $\{v, u'\} \in M$

- With the definition of *weak stability* an edge $\{u, v\} \notin M$ is considered blocking if $v \prec_u v'$ and $u \prec_v u'$.
- With the definition of *strong stability* an edge $\{u, v\} \notin M$ is considered blocking if $v \prec_u v'$ and $u \preceq_v u'$.
- With the definition of *super stability* an edge $\{u, v\} \notin M$ is considered blocking if $v \preceq_u v'$ and $u \preceq_v u'$.

The three problems Weakly Stable Matching, Strongly Stable Matching and Super Stable Matchings are defined as below. They only differ in the definition of blocking that is used.

Stable Matching
Input: An undirected graph $G = (V, E)$, and for each node $v \in V$ a preference list P_v, in which v weakly orders a subset $S \subseteq V(G)$
Question: Is there a matching $M \subseteq E(G)$ such that no edge $\{u, v\} \notin M$ is blocking.

Parameterized complexity. We will introduce basic concepts of parameterized complexity [Cyg+15].

A *parameterized problem* over an alphabet Σ is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, the number $k \in \mathbb{N}$ is called the *parameter*. The parameter k gives information about some aspect of the input instance. It can be chosen arbitrarily, although only some choices lead to useful results.

If we have a parameterized problem Q, we can then investigate the complexity of Q depending on the chosen parameter. A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed parameter tractable* if there exists a constant $c \in \mathbb{N}$, a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm which correctly decides for an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ whether $(x, k) \in Q$, using $f(k) \cdot |(x, k)|^c$ time. $|(x, k)| := |x| + k$ is defined as the size of the input instance. If the above applies, then we also say $Q \in FPT$.

Furthermore, a problem Q is fixed parameter tractable if and only if Q admits a *problem kernel*. A problem kernel for a paremeterized problem Q is a polynomial-time computable function $K : (\Sigma^* \times \mathbb{N}) \to (\Sigma^* \times \mathbb{N})$ such that for a computable function $h : \mathbb{N} \to \mathbb{N}$ and for all $(w, k) \in \Sigma^* \times \mathbb{N}$:

$$(w, k) \in Q \Leftrightarrow K(w, k) \in Q, \text{ and } |K(w, k)| \leq h(k).$$

In this thesis we will use the parameter *feedback edge number*. Given a graph $G = (V, E)$ the feedback edge number is the minimum number of edges that have to be deleted from G to become a tree.

Data Reduction Rules. For the introduction of the basic concepts of data reduction rules we again use definition from Cygan et al. [Cyg+15].

To develop a problem kernel, we use preprocessing steps. Here we solve relatively easy parts of the problem instance. By repeatitly applying varying preprocessing steps, we are able to gradually remove certain structures from the input graph and reduce the problem instance until we are left with the problem kernel. The preprocessing steps are

also referred to as *data reduction rules*. Given a paramterized problem $Q : \Sigma^* \times \mathbb{N}$ a data reduction rule is a polynomial-time computable function $\Phi : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$. We say that a data reduction rule is *correct* if it maps an instance (I, k) to an instance (I', k'), such that $(I, k) \in Q$ if and only if $(I', k') \in Q$.

In the development of data reduction rules we will often use the phrase of *deleting* a node $v \in V(G)$. Here, implicitly we not only delete v itself, but we also delete all edges incident to v. Furthermore, we delete v from all preference lists. If we delete an edge $\{u, v\} \in E(G)$, then we also not only delete the edge itself, but we also delete u from v's preference list and v from u's preference list.

Chapter 3

Strongly Stable Matching on Trees

In Chapter 1, we already mentioned that strongly stable matchings can be found in $\mathcal{O}(nm)$ time in general graphs [Kun16]. However, we can identify special cases where the problem can be solved much faster. One of these is the special case in which the input graph is a tree. As we will later see, computing a strongly stable matching in a n-node tree only takes $\mathcal{O}(n)$ time. But what causes such a big difference between general graphs and trees resulting in hugely different running times? One straightforward answer to this question could be that the *feedback edge number* makes the difference. The feedback edge number of a graph G is the number of edges $e \in E(G)$ that would have to be removed from G to obtain a tree or forest. Thus, the feedback edge number of a tree is naturally zero. The more different a graph-structure is from that of a tree, the higher its *feedback edge number* is. Hand in hand with this goes the number of cycles in a graph. It is easy to recognize that a tree contains no cycles. This makes the formulation of an algorithm for finding a strongly stable matching in a tree much easier.

In this chapter we will get acquainted with the topic of the Strongly Stable Matching problem. We will learn what to pay attention to when developing algorithms and data reduction rules to find a strongly stable matching. Therefore, we develop a linear-time algorithm for finding a strongly stable matching in the special case of a feedback edge number of zero, i.e., a tree or forest. This is relatively easy in comparison to the later development of data reduction rules. The goal of this chapter is to prove the following theorem:

Theorem 3.1. Strongly Stable Matching *on trees can be solved in* $\mathcal{O}(n)$ *time.*

In the following we will present an algorithm proving Theorem 3.1. As input the algorithm gets a tree G and a preference list for each node $v \in V(G)$, which weakly orders all neighbors $N(v)$. We fix some node to be the root. The algorithm consists of two phases. In the first phase, we traverse nodes in bottom up order. For every node of a level, we make a case distinction depending on the first preference of the node. We distinguish between three cases, whether one child, several children or only its parent is the nodes first preference. Depending on which case holds for a node we can directly add edges to the matching, immediately delete edges, remember that certain nodes have to be

part of a matching edge or determine that a strongly stable matching does not exist. We will see that after the completion of the first part of the algorithm the remaining graph has to have particular characteristics. These characteristics enable us to formulate the second phase in which the algorithm iterates over the remaining nodes in reverse order compared to the first phase. By the time we look upon a node during the second phase, the node has no parents itself. We make a case distinction, depending on whether the node has children or not and either add an edge to the matching or not. In this fashion we gradually delete all remaining nodes in the reduced graph and complete the strongly stable matching, if its exists.

As already mentioned we iterate over the levels of the graph G during the algorithm. Thus, we need to store the levels, such that each node is contained in exactly one level i, where $i \in \{1, 2, ..., \text{height}(G)\}$. The algorithm computes a strongly stable matching M of the given graph, if it exists. Initially we have $M = \emptyset$.

Phase 1

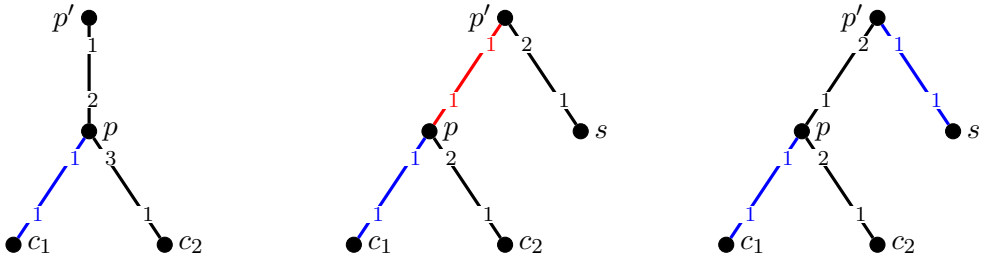
In the first phase, we repeat the following steps for every node in bottom up order, starting at the second highest level (level(height(G))) until we reach the second lowest level (level(2)). Looking at node $p \in \text{level}(i)$, let $C(p)$ be the set of children of p. As we will later see, by the time we look at a node p, every $c \in C(p)$ is a leaf. We make a case distinction, depending on the preference list of p.

Case 1.1: The first node in the preference list of p is one of its children $c \in C(p)$ and no other child is tied in first position. As c is a leaf, it only has p in its preference list. Thus, the edge $\{p, c\} \in E(G)$ has to be part of any strongly stable matching. Otherwise $\{p, c\}$ would always be a blocking edge, as p does not prefer any node to c and c prefers p to having no partner. Let $p' \in V(G)$ be the parent of p. Next we make a case distinction on whether there are other nodes tied in first position of p's preference list (see Figure 3.1 for a visualization).

Case 1.1.1 The first preference of p only consists of the child c_1. Thus, p' is not part of the first preference of p. In this case we add $\{p, c_1\}$ directly to M and delete c_1 and p. We also delete p from the preference list of its parent p' and the preference lists of all siblings of c_1. If the preference list of a node u becomes empty, then the node gets deleted. The only exception to this occurs when u is in the list F of nodes that have to be matched in M for it to be strongly stable. This list will get important in Case 1.1.3 and Case 1.3.2. If u is contained in F, then a strongly stable matching does not exist.

Case 1.1.2 The first preference of p consists of one child c_1 and its parent p' and the first preference of p' only consists of p. Hence, we would also have to take the edge $\{p, p'\} \in E(G)$ into the matching. Otherwise it would be a blocking edge, as p' would prefer p to all its other possible partners and p is indifferent between p' and c_1. Therefore, in this case a strongly stable matching does not exist.

Case 1.1.3 The first preference of p consists of one child c_1 and its parent p', and the first preference of p' does not only consist of p, or p is not the first



Case 1.1.1 Case 1.1.2 Case 1.1.3

Figure 3.1: Three examples of subtrees showing Case 1.1. The number on the edge $\{u, v\}$, which is closer to u indicates the position of v in the preference list of u. Red edges are blocking, in the case that the blue edge is added to the matching. We could also consider adding the red edge to the matching, in which case the blue edge would be blocking.

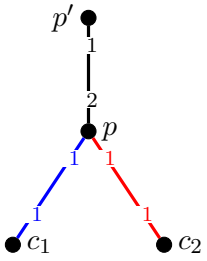


Figure 3.2: An example of a subtree showing Case 1.2. The number on the edge $\{u, v\}$, which is closer to u indicates the position of v in the preference list of u. Red edges are blocking, if the blue edge is added to the matching. We could also consider adding the red edge to the matching, in which case the blue edge would be blocking.

preference of p'. In this case a strongly stable matching may exist. We then add $\{p, c_1\} \in E(G)$ to M and delete c and p. Afterwards, we delete p and all nodes to which p' prefers p from the preference list of p'. We also have to remember p'. Thus we add p' to a list F, where we store all nodes that must be matched at the end. If we recognize in the second phase that p' has no partner, then we know that $\{p, p'\} \in E(G)$ is a blocking edge. We also delete p nodes from the preference lists of all siblings of c_1. If the preference list of a sibling node becomes empty, then we can delete it, unless this sibling node is contained in F, in which case a strongly stable matching does not exist.

Case 1.2: At least two children of p are tied as its first preference (see Figure 3.2 for visualization). Let $\{c_1, ..., c_\ell\} \subseteq C(p)$ be those children. Then there exists some $i \in \{1, ..., \ell\}$ with $\{p, c_i\} \notin M$. Hence edge $\{p, c_i\}$ is blocking, as c_i prefers p to having no partner and p does not prefer any nodes to c_i, as c_i is part of the first preference of p. Hence, in this case we instantly know that a strongly stable matching does not exist.

Case 1.3: The first preference of p is its own parent $p' \in V(G)$. In this case, at this

Case 1.3.1 Case 1.3.2

Figure 3.3: Two examples of subtrees showing Cases 1.3.1 and 1.3.2. The number on the edge $\{u, v\}$, which is closer to u indicates the position of v in the preference list of u. The ranking i of p in the preference list of p' could be any number and is one criterion that decides in Case 1.3.2 if a strongly stable matching exists. In Case 1.3.1, if $\{p', p\} \notin M$, then $\{p, c_1\} \in M$. In Case 1.3.2, if $\{p', p\} \notin M$, then either $\{p, c_1\}$ or $\{p, c_2\}$ is a blocking edge.

moment we still do not know which edge to add to the matching. Either $\{p, p'\} \in M$ or the edge between p and its most preferred child is contained in M (see Figure 3.3 for a visualization).

Case 1.3.1: The second preference of p consists of exactly one of its children. At this point in the algorithm, we do not add any edges to the matching and set all children of p to inactive. This means that when looking at the next lower level$(i-1)$, then the children of all nodes in in that level seem to have no children themselves and are treated as if they were leaves. After the first phase is finished, all inactive nodes will be activated again.

Case 1.3.2: The second preference of p consists of at least two of its children. Let $\{c_1, ..., c_\ell\} \subseteq C(p)$ be those children. For a strongly stable matching to exist, the edge $\{p, p'\} \in E(G)$ has to be part of the matching. Otherwise there would exist a node $c_i \in \{c_1, ..., c_\ell\}$ such that $\{p, c_i\}$ is blocking, as c_i prefers p to having no partner at all and p is tied between $\{c_1, ...c_\ell\}$. Hence, in this case we add p to the list F and delete all edges $\{p, c\}$ for all $c \in C(p)$.

If at any point in Phase 1 we find out that a strongly stable matching does not exist, then we stop. After going through all levels and reaching the root, we start the second phase. After the first phase the following statements apply:

Let G' be the graph after Phase 1.

Observation 3.2. *Graph G' is a forest.*

Proof. In Phase 1 nodes are deleted, which can destroy the tree-structure. As nodes or edges are never added, G' is a forest. □

Even though G' is a forest and therefore levels are not defined anymore, our data structure still saves each node in the level it had in G, which will become important in the second phase. Furthermore we refer to the nodes whose parents were deleted in G as roots in G'.

Lemma 3.3. *Every node $v \in V(G')$ that has a parent node either has no children, or prefers its parent node to all its children.*

Proof. If a node prefers exactly one of its children, then Case 1.1.1 applied and it was deleted. If it prefers several of its children, then Case 1.2 applied and a strongly stable matching does not exist. If the first preference of a node consists of one child and the parent node, then either Case 1.1.2 applied, in which case a strongly stable matching does not exist, or Case 1.1.3 applied and the node was deleted along with that child node. The only possibility for v still to exist is Case 1.3 which means that v prefers its parent node. □

Lemma 3.4. *The root node $r \in V(G)$ was deleted and hence $r \notin V(G')$.*

Proof. The only possibility for r not to be deleted would be that r does not strictly prefer one child to all the others. Then Case 1.2 would apply and a strongly stable matching does not exist, in which case the algorithm stops. □

Lemma 3.5. *Let r' be a root node of a connected component of G'. Either r' has no children, or it strictly prefers one of its children to all the other.*

Proof. If at least two children of r' are tied in first position, then Case 1.2 was applied in Phase 1. Thus a strongly stable matching does not exist. If at least two children of r' are tied in second position, then Case 1.3.2 was applied and all children are deleted from the preference list of r'. Hence, if r' does not prefer one child to all the other, then r' does not have children in Phase 2. The only possibility for r' to still exist and have children in Phase 2 is if Case 1.3.1 applied in Phase 1. In this case, node r' strictly prefers one of its children to all the others. □

Lemma 3.6. *If we delete a root r' of a connected component of G', then Lemma 3.5 still holds for the root nodes c of the consequently newly arising connected components.*

Proof. After deleting the root r', all its children $c \in C_{G'}(r')$ become root nodes themselves. We have to show, that if c has children, then it strictly prefers one of them to all other. As all $c \in V(G')$ are part of the graph after Phase 1, Lemma 3.3 applies for all nodes c. Hence the now deleted r' was the first preference of all nodes c. Therefore, Case 1.3 applied for c in Phase 1. If Case 1.3.1 applied, then c prefers exactly one of its children c_1. Futhermore, node c and c_1, as well as the edge $\{c, c_1\}$ were not deleted in Phase 1 and still exist in G'. If Case 1.3.2 applied, then c did have at least two of its children as second preference, but all edges $\{c, c_1\}$ were deleted, for all $c_1 \in C_G(c)$. Hence, in this case c does not have children in G'. □

The above lemmas enable us to formulate the second phase. There we start looking at the root node of a connected component. Lemma 3.3 admits an easy case distinction. In the first case the root node we are currently looking at has children and in the second case it does not have children. Depending on whether the root node has children we can then either add an edge to the matching or not. Moreover, in some cases we are able to detect if no strongly stable matching exists in the graph. Afterwards, we can delete the root node and thus produce more root nodes, over which we also iterate. Lemma 3.6

enables us to handle every root node we iterate over in the same way. Thus, we gradually iterate over all nodes of the remaining tree and also gradually remove the whole tree, while completing the strongly stable matching. We will now describe the second phase of the algorithm in detail.

Phase 2

First all nodes are activated again. The second phase starts at the lowest level, where nodes exists, i.e., where not all nodes are deleted. This cannot be the lowest level of G, as Lemma 3.4 states. Let i be the level we are currently looking at. Every node $v \in \text{level}(i)$ is regarded a root r of a tree. For every root $r \in V(G')$ we do the following:

Case 2.1: Node r has no children. This means that either r is a leaf in G or all children were deleted from its preference list in Phase 1. Let $v \in C_G(r)$ be a child of r in the original tree G. Node v was either deleted from the preference list of r because it is already matched to one of its own children, or because r prefers another matched child $u \in C_G(r)$ to v. The latter can only happen if Case 1.1.3 applied to u in Phase 1, in which case u is indifferent between its matched partner and r.

Case 2.1.1: If $r \in F$ where F is the list of nodes that have to be matched in M, then there exists a blocking edge $\{r, v\} \in E(G)$, where $v \in C_G(r)$ is part of a matching edge. This scenario is possible because of Case 1.1.3. In this case a strongly stable matching does not exist.

Case 2.1.2: If $r \notin F$, then r will not be part of a matching edge, but also not part of a blocking edge. Node r can be deleted.

Case 2.2: Node r has children. As r prefers one child $v \in C_{G'}(r)$ to all other children, as shown in Lemma 3.5 and Lemma 3.6, and every child prefers r to their own children, we add the edge $\{r, v\}$ to the matching M and delete r and v. We also delete r from the preference lists of all children of r and v from the preference lists of all children of v.

After finishing the steps above for all nodes in one level, they are either all deleted (Case 2.1.2 and Case 2.2) or a strongly stable matching does not exist (Case 2.1.1). We continue with the next higher level in which nodes exist. As all nodes in the lower levels were deleted, all nodes in this new level are now roots. When all nodes $v \in V(G')$ are deleted, then the algorithm has ended and M is a strongly stable matching in G.

To show Theorem 3.1 we now have to proove that the running time of the above algorithm is $\mathcal{O}(n)$. Let $G = (V, E)$ be the input graph. Graph G is a tree. In Phase 1 of the algorithm we look at all nodes $p \in V(G)$ bottom up, apart from the nodes at the heighest level.

The running time of Case 1.1.1 depends on the degree of p, as we delete p from the preference list of all its neighbors. The running time of Case 1.1.3 depends on the neighborhood of p's parent $p' \in V(G)$. It is possible to implement this in such a way that Phase 1 still has a linear running time, which we will explain now. We look in parallel at p and all its siblings s, that have one of their children and their parent as first

preference. We call this subset of sibling S. If p' has only one child as first preference, then Case 1.1.2 applies and a strongly stable matching does not exist.

Else we determine the node between p and all $s \in S$, which p' likes best and delete all others from the preference list of p'. Thus we have to look only once at the neighborhood of p' for all children where Case 1.1.2 or 1.1.3 applies. Hence, the running time is linearly dependent on the degree of p'.

Case 1.2 has a constant running time, as in this case a strongly stable matching does not exist. The running time of Case 1.3. is dependent on the number of children of p, and thus the degree of p as we either set all children to inactive (Case 1.3.1), or delete the edges $\{p, c\}$, for $c \in C_G(p)$ (Case 1.3.2).

For each node p exactly one of the cases described in Phase 1 holds. The running time of the cases is either constant, or dependent on the degree of p or its parent p'. But if it is dependent on the degree of p', then we will only have to look at p' once for all its children. This leads to a total running time of $\mathcal{O}(\sum_{p \in V(G)} \deg_G(p)) = \mathcal{O}(m) = \mathcal{O}(n)$ in Phase 1.

Phase 2 has a running time of $\mathcal{O}(n)$. First all nodes are activated, which takes $\mathcal{O}(n)$ time. In Case 2.1 we only have to check whether the node we are currently processing is contained in F. It is possible to implement F such that this takes constant time. In Case 2.2, where we add an edge $\{u, v\}$ to the matching, the running time is dependent on the number of children of u and v. As each node is at most child to one other node, this leads to a running time of $\mathcal{O}(m) = \mathcal{O}(n)$ as the graph is a tree. Thus the total running time of Phase 2 is $\mathcal{O}(n)$.

As both Phase 1 and Phase 2 have a running time of $\mathcal{O}(n)$ the total running time of the algorithm also is $\mathcal{O}(n)$. This completes the proof of Theorem 3.1

Chapter 4

Data Reduction Rules

The final goal of this thesis is to find a linear-size, linear-time computable kernel for Strongly Stable Matching parameterized by the feedback edge number. We achive this by confining the number of degree-zero, degree-one and degree-two nodes in the input graph. In this chapter, we will present data reduction rules deleting all degree-zero and degree-one nodes in a graph and for reducing all subgraphs of degree-two nodes to constant length. Working towards those data reduction rules, we will first present general data reduction rules that have to be applied before the data reduction rules for degree-one nodes and degree-two nodes can be applied. In each section we will first present the data reduction rules and the proofs of their correctness and at the end of each section we will then show that the data reduction rules can be applied exhaustively in linear time.

In the development of the data reduction rules we face two problems, which will lead us two the formulation of an annotated version of Strongly Stable Matching. As already mentioned, we have to find data reduction rules that delete all degree-one nodes. However, we often cannot just delete a node, as this could delete blocking edges. Consider for example the following case, which is also presented in the left graph of Figure 4.1. Let $G = (V, E)$ be a graph containing a degree-one node $u \in V$. Node u's only neighbor and first preference is node v. Node v has one other neighbor, node w. Furthermore, node v is indifferent between both of its neighbors. Node w has more neighbors than just v. If a strongly stable matching exists in G, then it is obvious that $\{u, v\}$ has to be part of it, as it would be blocking otherwise. But we still cannot just delete u and v, as the edge $\{v, w\}$ is also blocking if w does not get matched to a partner it likes at least as much as v. Thus, we need the means to remember that while we delete u and v, node w must also be matched, as it would be blocking in G otherwise. To this end, we introduce an annotated version of Strongly Stable Matching, which we call Annotated Strongly Stable Matching. In Annotated Strongly Stable Matching additionally to the input graph G and the preference lists for each node, we have an input set $F \subseteq V$ of nodes that have to be matched in the matching in G. Moreover, we will need one more additional input, as illustrated in the following extended scenario, which is presented in the right graph of Figure 4.1.

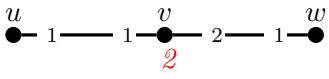
Again, we look at a degree-one node $u \in V$, which has only one preference and neighbor $v \in V$. This time, node v has three neighbors, the nodes $u \in N(v)$ and $w_1, w_2 \in$

Figure 4.1: Two examples showing why Annotated Strongly Stable Matching can be useful when reducing a graph. In graph G_1 the blue edge $\{u, v\}$ has to be part of any strongly stable matching in the graph, as it would be blocking otherwise. The red edge $\{v, w\}$ is blocking if w is not matched to another node it likes at least as much as v. In graph G_2 the black edge $\{v, w_1\}$ is v's preferred choice of being matched. If $\{v, w_1\}$ cannot be in the matching, then the blue edge $\{u, v\}$ must be in the matching and w_1 must be matched to a neighbor it likes at least as much as v. The red edge $\{v, w_2\}$ cannot be in the matching, independent of whether v can be matched to w_2 or not. However, it could become a blocking edge. The dots indicate that this is a subgraph, and the graph continues at those points.

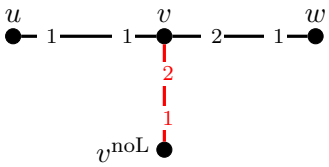
$N(v)$. Node v, however, does not have u as its first preference. Instead, node w_1 is the first preference of v. Nodes u and w_2 are both second preferences of v. Now, edge $\{u, v\}$ does not necessarily have to be in the matching. If edge $\{v, w_1\}$ is in the matching, then the edge $\{u, v\}$ is not blocking, as v prefers w_1 to u. If, however, the node w_1 is not matched to v, then we know that the edge $\{u, v\}$ has to be in the matching. If $\{u, v\}$ is not in the matching and the node v is for example matched to w_2, then the edge $\{u, v\}$ is blocking, as v is indifferent between u and w_2. In this scenario, if node w_1 is not matched to v and node v is matched to u, then the matching can only be strongly stable if w_2 is also matched to another neighbor it likes at least as much as v. Otherwise, edge $\{v, w_2\}$ would be blocking. However, we cannot simply add w_2 to the set F of nodes that have to matched, as the edge $\{v, w_2\}$ is not blocking if v is matched to w_1 and thus node w_2 does not necessarily has to be matched. Thus, if we want to delete the node u we need to remember, that v formerly had a neighbor as its second preference. Should we then realize that v cannot be matched to w_1, then we know that w_2 has to be matched to a node it likes at least as much as v. Hence, we introduce a set $L \subseteq V$ of labeled nodes. A labeled node $v \in L$, carrying the label $\ell_v \in \mathbb{N}$, is treated as if it had one additional neighbor, which is at the ℓ_v^{th} position in v's preference list. We will now define a graph called $\text{noL}(G)$ where all these additional neighbors actually exist. An example is provided in Figure 4.2.

Definition 4.1. Let G be a graph, and $L \subseteq V(G)$ a set of labeled nodes, and for each node $v \in V(G)$ let $P_G(v)$ be the preference list of v. For each labeled node $v \in L$ let $\ell_v \in \mathbb{N}$ be the label of v. W.l.o.g we say that $V(G) = \{v_1, v_2, ..., v_n\}$. Then, the graph $\text{noL}(G)$ is:

$$\text{noL}(G) := (V(G) \cup \{v_i^{\text{noL}} \mid v_i \in L\}, E(G) \cup \{\{v_i, v_i^{\text{noL}}\} \mid v_i \in L\})$$



Graph G



Graph noL(G)

Figure 4.2: An example showing a graph G and its corresponding graph noL(G). Graph G contains one labeled node v. Node v is labeled with *2*. Thus, in noL(G) node v has one additional neighbor and second preference v^{noL}. The new node v^{noL} only has one preference, which is node v. In graph noL(G) node v is not labeled anymore. As both u and w are not labeled in G, they get no additional neighbors in noL(G) and their preference lists remain the same.

Furthermore, the preference list $P_{\text{noL}(G)}(u)$ for each node $u \in V(\text{noL}(G))$ is:

$$P_{\text{noL}(G)}(u) := \begin{cases} P_G(u), & \text{if } u \in V(G) \setminus L, \\ \pi_{1,\dots,\ell_u-1}(P_G(u)) \circ ((\{v_i^{\text{noL}}\}) \cup \pi_\ell(P_G(u))) & \text{if } u = v_i \in L, \\ \circ\, \pi_{\ell_u+1,\dots,|P_G(u)|}(P_G(u)), & \\ (v_i), & \text{if } u = v_i^{\text{noL}}. \end{cases}$$

When looking at labeled nodes v, we often have to compare v's neighbors to its label. Let P_v be the preference list of v and u be a neighbor of v. Let $\pi_i(P_v) = Q_i$ with $u \in Q_i$ be the set of nodes in v's preference list in which u is a part of. With this we know that u is at the i^{th} place in v's preference list. From this point onward we will say v prefers ℓ_v to u if $\ell_v < i$. Accordingly, we say v prefers u to ℓ_v if $u < \ell_v$ and we say v is indifferent between u and ℓ_v if $\ell_v = i$.

With the definition of G_{noL}, we can now introduce *annotated strongly stable matchings*:

Definition 4.2. Let G be a graph, let $F \subseteq V(G)$ be a set of nodes and let $L \subseteq V(G)$ be a sets of labeled nodes, where ℓ_v is the label for each node $v \in L$. Furthermore, for each node $v \in V(G)$ let $P_G(v)$ be the preference list of v. A matching $M \subseteq V$ is an annotated strongly stable matching if

1. $M_{\text{noL}} := M \cup \{\{v, v^{\text{noL}}\} \mid v \in L : \forall e \in M : v \notin e\}$ is strongly stable in noL(G), with preference lists $P_{\text{noL}(G)}(v)$ for each node $v \in V(\text{noL}(G))$, and
2. each node $v \in F$ is matched in M_{noL}

With the definition of annotated strongly stable matching available we can now easily define the Annotated Strongly Stable Matching problem:

Annotated Strongly Stable Matching
Input: An instance $(G = (V, E), F, L, P)$, where G is a graph with a preference list $P_v \in P$ for each node $v \in V$, and two sets of nodes $F, L \subseteq V$, with labels $\ell_v \in \mathbb{N}$ for each node $v \in L$.
Question: Does an annotated strongly stable matching exist in G?

We may observe that in the formal definition of annotated strongly stable matching the influence of labeled nodes is not very intuitive. Nevertheless, we have to consider the labeling in the proof of correctness for all the following data reduction rules. Hence, it would be highly practical to have a more intuitive approach on the influence of labels in an annotated strongly stable matching. Taking a closer look at the second point of Definition 4.2 we can see that in the extended graph noL(G), as defined in Definition 4.1, each labeled node $v \in L$ has to be matched. A node v is either already matched in the matching M in the input graph G, or v gets matched to its new neighbor v^{noL}, which only exists in noL(G), and represents the label of v in G. This forms the matching M_{noL} in noL(G). A deciding factor in understanding Definition 4.2 is that M_{noL} must be strongly stable in noL(G). Thus, if a labeled node v is matched in the annotated strongly stable matching M in G it must be matched to a node it prefers to its label ℓ_v. If v were matched to a node u it did not prefer to its label, then the edge $\{v, v^{\text{noL}}\}$ would be blocking in noL(G), as v^{noL} is at the ℓ^{th} place in v's preference list and the node v^{noL} is a degree-one node. If v is not matched in the annotated strongly stable matching M in the input graph G, then it gets matched to the additional neighbor v^{noL} in noL(G). In order for the matching M_{noL} to be strongly stable each neighbors $u \in N_G(v)$, which v likes exactly as much as v^{noL} must be matched to a neighbor, it likes at least as much as v. Otherwise, the edge $\{u, v\}$ would be blocking. The only difference between the matching M in G and the matching M_{noL} in noL(G) are the edges between labeled nodes v and their new neighbors v^{noL}. Thus, we know that if a labeled node v is not matched in M, then each of v neighbors which v likes exactly as much as its label must be matched to a node it likes at least as much as v in M. We can summarize this in the following observation:

Observation 4.3. *Let $(G = (V, E), F, L, P)$ be a an instance of* Annotated Strongly Stable Matching*, where $L \subseteq V$ is the set of labeled nodes, and M is an annotated strongly stable matching in G. Each node $v \in L$, labeled with label ℓ_v is either*

- *matched, to a node it prefers to ℓ_v, or*
- *is not matched and every neighbor u of v, which v likes exactly as much as its label, must be matched to one of its neighbors which it likes at least as much as v.*

First we formulate a simple data reduction rule to remove degree-zero nodes. It is obvious that any node of degree zero cannot be matched or produce blocking edges, as they do not have any incident edges. Different from the other data reduction rules we develop in this section, this data reduction rule is completely independent from the other data reduction rules, i.e., it can be applied at any time and does not have to be applied for any of the other data reduction rules to be applicable. Thus, we will also show proof of its linear running time immediately. While this data reduction rule might seem unnecessary we still need it to formally show that we are able to confine the number of nodes in a graph.

Reduction Rule 4.1. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *and let $v \in V$ be a node with $\deg(v) = 0$. Delete node v.*

Lemma 4.4. *Reduction Rule 4.1 is correct and can be applied exhaustively in $\mathcal{O}(n)$.*

Proof. Let M be an annotated strongly stable matching in the input graph G and let G' be the reduced graph. The only nodes that exist in graph G but not in G' are degree-zero nodes. As degree-zero nodes are not incident to any edges, they cannot be matched or produce blocking edges. Thus, we can remove any degree-zero node without influencing the matching. Hence, matching M is annotated strongly stable in G if and only if M is also annotated strongly stable in G'. Removing a node takes constant time. Thus, Reduction Rule 4.1 can be applied exhaustively in $\mathcal{O}(n)$ time. □

Looking at Observation 4.3 it is obvious that if the label of a node v is a higher number than the last preference of v or *1*, then the influence of this labels on the node v is restricted. We can show that in these two case, it is possible to immediately remove some labels. To do so we apply the following data reduction rule.

Reduction Rule 4.2. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching. *For each labeled node $v \in L$, let $\text{last}(v)$ be the last position in v's preference list and ℓ_v be v's label.*

(a) If $\text{last}(v) < \ell_v$, then delete the label of v.

(b) If $\ell_v = 1$, then delete v. Moreover, for each $u \in N_G(v)$ which is a first preference of v delete all neighbors of u it likes less than v and add u to F.

Lemma 4.5. *Reduction Rule 4.2 is correct.*

Proof. Let M be an annotated strongly stable matching in the input graph G and $F(G) \subseteq V(G)$ be the set of nodes that have to be matched in M, and let G' be the reduced graph, where $F(G') \subseteq V(G')$ is the set of nodes that have to be matched. Let $v \in V(G)$ be the node for which the reduction was applied. Node v is labeled with label ℓ_v. First we look at the case that $\text{last}(v) < \ell_v$. If v is matched in M, then by Observation 4.3 it must be matched to a node it prefers to its ℓ^{th} position in its preference list. Thus, the matching M is also annotated strongly stable in M', where v is not labeled. If v is unmatched in M, then all neighbors $u \in N_G(v)$ must be matched to a node they prefer to v, as otherwise the edge $\{u, v\}$ would be blocking as v prefers all its neighbors to its label. Hence M is also annotated strongly stable in G'.

Let M' be an annotated strongly stable matching in G'. If v is matched in M', then adding a label ℓ_v, with $\text{last}(v) < \ell_v$ to v does not influence the annotated strong stability of M'. If v is matched in M', then v still prefers its matching partner to its label in G. If v is not matched in M', then all neighbors $u \in N_G(v)$ must be matched to a node they prefer to v, as otherwise the edge $\{u, v\}$ would be blocking, as v prefers all its neighbors to not being matched at all. This still holds if v is labeled with a label $\text{last}(v) < \ell_v$. Hence M' is also annotated strongly stable in G.

Now we look at the case that $\ell_v = 1$. This results in v not having any neighbors it prefers to ℓ^v. Thus, node v cannot be matched in M. Let $u \in V(G)$ be a first preference of v. For the edge $\{u, v\}$ not to be blocking, node u must be matched to a node it likes at least as much as v, as v is indifferent between not being matched (because of its label)

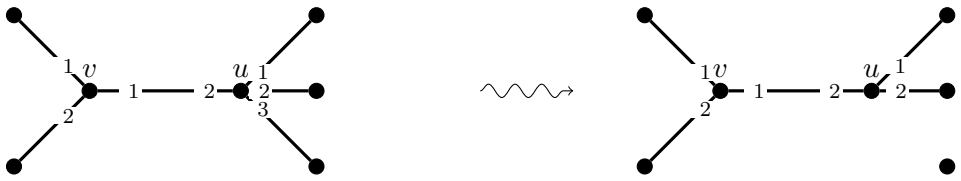


Figure 4.3: Examples showing Reduction Rule 4.4. Node v has u as first preference. Hence u cannot be matched to any node it prefers v to, as otherwise the edge $\{u, v\}$ would be blocking.

and being matched to u. Thus, all nodes $w \in F(G')$ are matched to a node that also exists in G' in M and M is also annotated strongly stable in G'.

Let M' be an annotated strongly stable matching in G'. As v does not exist in G' it cannot be matched in M'. Let u be a first preference of v in the graph G. Then, it holds that $u \in F(G')$. Hence, node u is matched in M'. Node u is also matched to a node it likes at least as much as v in the graph G as all other neighbors of u were deleted in the reduction. Thus, the matching M' is also annotated strongly stable in G. □

Reduction Rule 4.2 can be seen as preprossessing of the input graph, as we delete labels in simple cases, where we can immediately see that they do not have any influence on the annotated strong stability of any matching. We will now develop an additional data reduction rule, which can also be viewed as preprossessing.

Reduction Rule 4.3. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching. *Let $v \in L$ be labeled with label ℓ_v. For all neighbors $u \in N_G(v)$: If v prefers ℓ_v to u, then delete the edge $\{u, v\}$.*

Lemma 4.6. *Reduction Rule 4.3 is correct.*

Proof. Let M be an annotated strongly stable matching in the input graph G and let G' be the reduced graph. Let $v \in L(G)$ be labeled with label ℓ_v, and let $u \in N_G(v)$ be a neighbor of v, which v likes less than ℓ_v. Then, node v cannot be matched to u in the matching M, as v prefers not being matched to being matched to u. Thus, the matching M is also annotated strongly stable in G', where all edges $\{u, v\}$ are deleted during the reduction.

Let M' be an annotated strongly stable matching in G'. In G' no edges $\{u, v\}$ with v preferring ℓ_v to u exist and therefore cannot be in the matching M'. In G, where those edges exist, they cannot be blocking, as v prefers not being matched to being matched to such nodes u, due to v's label. Hence, the matching M' is also an annotated strongly stable matching in G. □

After having applied Reduction Rule 4.3 exhaustively, we can apply a data reduction rule that deletes some further edges that cannot be in a matching. Reduction Rule 4.4 is visualized in Figure 4.3.

Reduction Rule 4.4. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *where Reduction Rules 4.2 and 4.3 are not applicable. If there exists an edge $\{u, v\} \in E$ where v's first preference is u, then delete all edges $\{u, w\} \in E$, where $v \prec_u w$, i.e., delete all nodes, which u likes less then v, from the preference list of u. If u is labeled with a label it likes less than v, then delete u's label.*

Lemma 4.7. *Reduction Rule 4.4 is correct.*

Proof. Let M be an annotated strongly stable matching in the input graph G and let G' be the reduced graph. All edges that G has and G' does not have cannot be part of an annotated strongly stable matching: In the reduction only edges $\{u, w\} \in E(G)$ with $v \prec_u w$ are deleted. Assume $\{u, w\} \in M$. Then the edge $\{u, v\}$ is blocking, as u prefers v to its partner w and v does not prefer the node it is matched to in M to u. Thus, such edges $\{u, w\}$, where $v \prec_u w$ cannot be part of an annotated strongly stable matching.

As we want to show the correctness for Annotated Strongly Stable Matching, we have to consider the set $F(G)$ of nodes that have to be matched in M. The set $F(G)$ does not change during the reduction and therefore does not influence whether M is also annotated strongly stable in G'. We also have to consider that u or v could be labeled. As v has node u as its first preference, but can only labeled with numbers higher than *1* due to *ReductionRule* 4.2, a possible labeling of v does not influence Reduction Rule 4.4. If u is labeled with label ℓ_u, then u must like v at least as much as ℓ_u. Otherwise, Reduction Rule 4.3 would be applicable.

First we will look at the case that u is indifferent between v and ℓ_u. In this case u cannot be matched to v. Moreover, as Reduction Rule 4.3 is not applicable, node u does not have any neighbors it likes less than ℓ_u and thus also v. Hence, in this case Reduction Rule 4.4 does not change the input at all and therefore is correct.

In the following we look at the case that u prefers v to ℓ_u. The label ℓ_u does not have any influence on the matching, as u would always be matched to v if it could not be matched to a node it prefers to v. Thus, it is possible for u to be matched to a node it prefers to ℓ_u. Hence, for the annotated strong stability of matching M it does not make a difference whether u is labeled with ℓ_u, or not. Consequently, if G has an annotated strongly stable matching M, then M is also annotated strongly stable in G'.

Now we show that if an annotated strongly stable matching exists in G', then in G an annotated strongly stable matching exists, too. As u prefers v to ℓ_u, the label of u gets deleted during the reduction and u is unlabeled in the reduced graph G'. In G', node u only has neighbors which it likes at least as much as v, as all other neighbors were deleted during the reduction. First, we show that if G' has an annotated strongly stable matching M', then u has to be matched in it. Assume u is not part of any edge in M'. Then the edge $\{u, v\} \in E(G')$ would be blocking, as u is part of v's first preference and u prefers v to not being matched at all. Thus, M' is also a strongly stable matching in G, as the only edges that G has and G' not has are edges $\{u, w\}$, where $v \prec_u w$. Due to the same reasoning as above, the set F of nodes that have to be matched does not influence this. □

With Reduction Rules 4.2 to 4.4 we have formulated all data reduction rules we need in preparation for the following section, where we present the data reduction rules deleting the degree-one nodes. However, we still need to show, that Reduction Rules 4.2 to 4.4 can be applied exhaustively in linear time. While this is easy for Reduction Rules 4.2 and 4.3, we need a slightly more complex approach to show the same for Reduction Rule 4.4.

Lemma 4.8. *Reduction Rules 4.2 and 4.3 can be applied exhaustively in $\mathcal{O}(n + m)$ time.*

Proof. Let $(G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching. In both Reduction Rules 4.2 and 4.3 we have to apply the data reduction rule on every labeled node. As $L \subseteq V$ we have a maximum of n labeled nodes. In Reduction Rule 4.2 determining whether the label ℓ_v of a node $v \in L$ is a *1* takes constant time. To determine whether $\ell_v > \text{last}(v)$ we have to compute the cardinality of v's preference list. The running time of this is dependent on the number of neighbors $\deg(v)$. In Reduction Rule 4.3, for each labeled node $v \in L$ we have to delete the edges to all neighbors of v, where v prefers its label ℓ_v to the neighbor. The running time of this is also dependent on the number of neighbors $\deg(v)$. Hence, for both Reduction Rules 4.2 and 4.3 the running time for applying them exhaustively is $\mathcal{O}(n + \sum_{u \in V(G)} \deg(u)) = \mathcal{O}(n + m)$. □

Looking at Reduction Rule 4.4 it is easy to see that applying it exhaustively using a brute-force implementation would lead to a running time of $\mathcal{O}(nm)$. First we search for edges $\{u, v\} \in E(G)$ where v's first preference is u. Then we delete all neighbors w of u, where $v \prec_u w$. However, u might have another neighbor v', which also has u as its first preference. If $v' \prec_u v$, then v' did not get deleted during the previous step. Thus, we would look at the edge $\{u, v'\}$ during a following iteration. In this iteration we would delete all neighbors w' of u, with $v' \prec_u w'$, amongst which is also v. It is obvious that this approach is ineffective. In the following we will see, that a slightly more sophisticated implementation leads to a much improved running time:

Lemma 4.9. *Reduction Rule 4.4 can be applied exhaustively in $\mathcal{O}(n + m)$ time.*

Proof. Let $G = (V, E)$ be the input graph for Annotated Strongly Stable Matching. For each $u \in V$ check all its neighbors. Let $A \subseteq N(u)$ be the neighbors of u that have u as their first preference. Amongst all nodes in A take one node $v \in A$ that u likes best. Then delete all edges $\{u, w\} \in E$, where $v \prec_u w$. Using this implementation, we only have to look one time at each u and two times at each of u's neighbors. Hence, we have a running time of $\mathcal{O}(n + \sum_{u \in V(G)} \deg(u)) = \mathcal{O}(n + m)$ for applying Reduction Rule 4.4 exhaustively. □

4.1 Degree-One Nodes

In Annotated Strongly Stable Matching we can distinguish between three general cases when looking at nodes $v \in V$ with $\deg(v) = 1$. Let u be the only neighbor of v. Then either (1) v is the single first preference of u, (2) node v is tied in first preference, or (3) v is not a first preference of u. In the first two cases, with the help of the list F of nodes that have to be matched, data reduction rules can be applied. The last case is not as simple. Here the labeling of nodes becomes of great importance.

Before applying the following data reduction rules for degree-one nodes, it is necessary to apply Reduction Rules 4.2 to 4.4 exhaustively. Afterwards, for each labeled node $v \in L$ its label ℓ_v will be the same as v's last preference in its preference list P_v, i.e., $\ell_v = \text{last}(v)$. Moreover, all edges $\{u, w\} \in E$, where u has another neighbor v which it prefers to w and v's first preference is u, are deleted. We will now look at Reduction Rule 4.5 which is visualized in Figure 4.4.

Figure 4.4: An example showing Reduction Rule 4.5. Node v has u as its only neighbor. Node v is the single first preference of u. As Reduction Rule 4.4 is not applicable the dotted edges were already deleted and cannot exist. Hence, the edge $\{u, v\}$ must be in the matching and nodes u and v can be deleted.

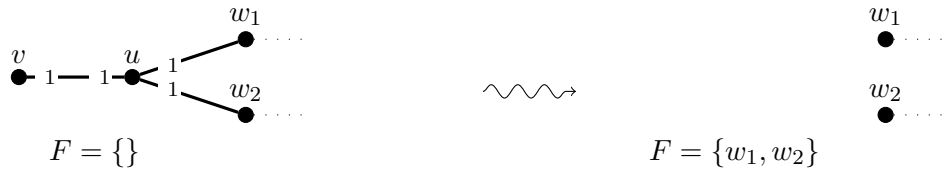


Figure 4.5: Examples showing Reduction Rule 4.6. Node v has u as its only neighbor. Node v is one of several first preferences of u. As Reduction Rule 4.4 is not applicable node u can only have first preferences. The edge $\{u, v\}$ must be in the matching. All other neighbors $w \in N(u) \setminus \{v\}$ must be matched to a node they like at least as much as u. Otherwise, $\{u, w\}$ would be blocking. The dots indicate that this is a subgraph, and the graph continues at those points.

Reduction Rule 4.5. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *where Reduction Rules 4.2 to 4.4 are not applicable. Let $v \in V$ be a degree-one node with neighbor $u \in V$. If node v is the single first preference of u, then delete the nodes u and v.*

Lemma 4.10. *Reduction Rule 4.5 is correct.*

Proof. Let G be the input graph, and G' be the reduced graph. Node v is the single first preference of u and u is the single neighbor of v. As Reduction Rule 4.4 was applied exhaustively before applying Reduction Rule 4.5, no edges $\{u, w\} \in E(G)$ with $v \neq w$ exist anymore. Hence, u's only neighbor is v and v's only neighbor is u. Reduction Rule 4.2 is also not applicable. Thus, neither u nor v can be labeled and we do not have to further consider labels in this proof. We now show that there exists an annotated strongly stable matching M in G if and only if there exists an annotated strongly stable matching M' in G'. Both u and v prefer each other to having no partner at all. Thus, $\{u, v\}$ has to be part of the matching M in G. As only node u and node v were deleted during the reduction it follows that $M \setminus \{\{u, v\}\}$ is an annotated strongly stable matching in G'. For an annotated strongly stable matching M' in G' it applies that $M' \cup \{u, v\}$ is an annotated strongly stable matching in G. □

The next data reduction rule will handle the second case of degree-one nodes. Here the only neighbor of the degree-one node has several first preferences. Thus, the degree-one node is tied in first preference. Reduction Rule 4.6 is visualized in Figure 4.5.

Reduction Rule 4.6. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *where Reduction Rules 4.2 to 4.4 are not applicable. Let $v \in V$ be a degree-one node with neighbor $u \in V$. If the first preference of u is v and*

at least one other node $w \in V$, then add all first preferences $w \in N(u) \setminus \{v\}$ of u to the set F of nodes that have to be matched and delete u and v.

Lemma 4.11. *Reduction Rule 4.6 is correct.*

Proof. Let G be the input graph for Annotated Strongly Stable Matching and let G' be the reduced graph. Let further $F(G) \subseteq V(G)$ be the input set of nodes that have to be matched. Let $v \in V(G)$ be a node in G with $\deg_G(v) = 1$ and neighbor u. Let $\{v, w_1, ..., w_\ell\} \in V(G)$ be the first preferences of u. Since Reduction Rule 4.4 is not applicable, no edges $\{u, x\}$ with $x \notin \{v, w_1, ..., w_\ell\}$ exist. Reduction Rule 4.2 is not applicable, too. Thus, neither u, nor v can be labeled and we do not have to further consider labels in this proof.

If an annotated strongly stable matching M exists in G, then the edge $\{u, v\} \in E(G)$ has to be part of it, as v prefers u to not being matched and u does not prefer any nodes to v. Each node $w \in \{w_1, ..., w_\ell\}$ has to be matched to a node it likes at least as much as u in M, as u is indifferent between v and w and otherwise $\{u, w\}$ would be blocking. Nodes u and node v are deleted during the reduction. Moreover, all nodes $w \in \{w_1, ..., w_\ell\}$ are contained in $F(G')$ the set of nodes that have to be matched in G'. Hence, $M \setminus \{\{u, v\}\}$ is an annotated strongly stable matching in G'.

Let M' be an annotated strongly stable matching in G'. If matching M' exists in G', then all nodes $w \in \{w_1, ..., w_\ell\}$ are matched, as they are contained in $F(G')$. In G all nodes $w \in \{w_1, ..., w_\ell\}$ have to be matched as well, as the edges $\{u, w\}$ would be blocking otherwise. Reduction Rule 4.4 was applied exhaustively beforehand and node u does not exist in G'. Thus, in M' all $w \in \{w_1, ..., w_\ell\}$ are matched to a node, which they like at least as much as u in G', but is not u itself. Hence, $M' \cup \{\{u, v\}\}$ is an annotated strongly stable matching in G. □

The only case left is that of degree-one nodes which are not the first preference of their only neighbor. This case will be handled in the following data reduction rule, which is also visualized in Figure 4.6.

Reduction Rule 4.7. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *where Reduction Rules 4.2 to 4.4 are not applicable. Let $v \in V$ be a degree-one node with neighbor $u \in V$ such that v is not the first preference of u. Let v be the ℓ^{th} preference of u. Delete the node v. If u has at least one other neighbor x as ℓ^{th} preference, then add the label ℓ to u.*

If u was already labeled with ℓ before, then delete the node v, delete all edges to nodes $w \in N(u)$, which are u's ℓ^{th} preference, delete the label of u and add u to F.

Lemma 4.12. *Reduction Rule 4.7 is correct.*

Proof. Let G be the input graph for Annotated Strongly Stable Matching and let G' be the reduced graph. Let $v \in V(G)$ be a node in G with $\deg_G(v) = 1$ and neighbor u. Furthermore, let $W \subseteq N_G(u)$ be the nodes which u prefers to v and let $X \subseteq N_G(u) \setminus \{v\}$ be the nodes that are tied with v in the preference list of u. As Reduction Rule 4.4 is not applicable, node u has no neighbors which it likes less than v. First we look at the case that node u is not labeled. Let M be an annotated strongly stable matching in G and M' an annotated strongly stable matching in G'.

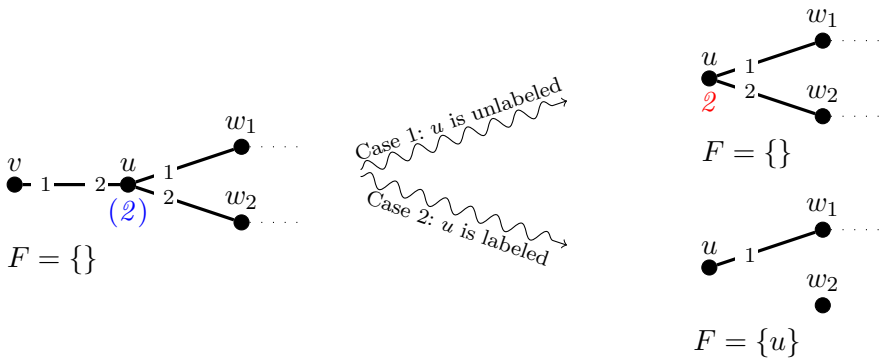


Figure 4.6: An example showing Reduction Rule 4.7. Node v has u as its only neighbor. Node v is not the first, but only the second preferences of u. There are two possible cases, depending on whether u is already labeled before Reduction Rule 4.7 is applied. In both cases we, delete the degree-one node v. In the case that u is unlabeled before Reduction Rule 4.7 is applied we add 2 as a label to u, as v is u's second preference. The set F of nodes that have to be matched remains unchanged. In the case that u is labeled before Reduction Rule 4.7 is applied we delete all neighbors of u which u likes exactly as much as its label. In this example this is node w_2. We also delete the label of u and add u to the set F of nodes that have to be matched. The dots indicate that these are subgraphs, and the graph continues at those points.

We will first look at the case that $X = \emptyset$. In that case, only v gets deleted. If $\{u, v\} \in M$, then all nodes $w \in W$ are matched to a node they prefer to u, since otherwise $\{u, w\}$ would be blocking, as u prefers w to v. Thus $M \setminus \{\{u, v\}\}$ is an annotated strongly stable matching in G'. If $\{u, v\} \notin M$, then there exists a node $w \in W$ such that $\{u, w\} \in M$. Thus, M is also an annotated strongly stable matching in G'. Looking at the other direction, if u is unmatched in M', then $M' \cup \{\{u, v\}\}$ is an annotated strongly stable matching in G. Otherwise M' itself is an annotated strongly stable matching in G.

Now we look at the case that $X \neq \emptyset$. Let ℓ be u's preference for v. Then all $x \in X$ also hold the preference ℓ in the preference list of u. In the reduction the label ℓ is added to the node u and v gets deleted. If $\{u, v\} \in M$, then all nodes $w \in W$ are matched to a node they prefer to u. Moreover all nodes $x \in X$ are matched to a node they like at least as much as u, as otherwise the edge $\{u, x\}$ would be blocking. Because of the labeling of u in G', the matching $M' = M \setminus \{\{u, v\}\}$ is an annotated strongly stable matching in G'. As u is unmatched in M', without the labeling, the edge $\{u, x\}$ would be blocking if x were indifferent between its partner in the matching and u. But due to the labeling, node u is treated as if it were already matched to a node holding the ℓ^{th} position in its preference list. Thus, node u is indifferent between being unmatched and being matched to an $x \in X$, and the edge $\{u, x\}$ is not blocking.

If $\{u, v\} \notin M$, then there exists a node $w \in W$ such that $\{u, w\} \in M$. Hence, no edge $\{u, x\}$, with $x \in X$ is blocking, as u prefers w to x. The same applies for the reduced graph. Thus, the matching M is also an annotated strongly stable matching in G'.

If u is unmatched in M', then $M' \cup \{\{u, v\}\}$ is an annotated strongly stable matching in G, as the only difference between G and G' is the node v and the edge $\{u, v\}$, which are not contained in G'. In G the node u is not labeled. Hence, edge $\{u, v\}$ would be blocking if it were not in the matching, as both u and v prefer each other to having no

partner at all. If u is matched in M', then it has to be matched to a node $w \in W$, as the labeling prevents it from being matched to any node $x \in X$. Hence, in this case the matching M' is also an annotated strongly stable matching in G, as u prefers its partner in M' to v for which reason the edge $\{u, v\}$ is not blocking in G.

In the above proof, we only considered the case that u was not already labeled before the application of Reduction Rule 4.7. Now we will look at the influence of labeling on Reduction Rule 4.7. Reduction Rules 4.2 to 4.4 are not applicable. Therefore, node v cannot be labeled and node u can only be labeled with last(u), which must also be v's position in the preference list of u. Let M be an annotated strongly stable matching in G. As u is labeled, edge $\{u, v\}$ cannot be in the matching M. However, for edge $\{u, v\}$ not to be blocking, either u must be matched to a node it prefers to v, or node v must be matched to a node it likes as least as much as u. But node v's only neighbor is u, so it is not possible for v to be matched in M. Thus, the only possibility is that u is matched to a node it prefers to v in M. Hence, we can delete v (as it cannot be matched), add u to F (as it must be matched) and delete all edges to neighbors of u it likes the same as v. As then neighbors with the placing ℓ in u's preference list do not exist anymore, we can also delete the labeling of u. The matching M must also be annotated strongly stable in the reduced graph G' and vice versa an annotated strongly stable matching M' in the reduced graph G' is also annotated strongly stable G. □

We still have to show, that applying Reduction Rules 4.5 to 4.7 exhaustively has a linear running time.

Lemma 4.13. *The running time of applying Reduction Rules 4.5 to 4.7 on a degree-one neighbor of a node u is linearly dependent on the number of edges deleted thereby.*

Proof. Let $(G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching and let $v \in V$ be a degree-one node. First we look at Reduction Rules 4.5 and 4.6. Here we delete v and its neighbor u. It is also necessary to delete u from all its neighbor's preference lists. In the case of Reduction Rule 4.6, we also add all first preferences of u to the list F. In both cases we delete the edges between u and all its neighbors. Hence, the running time of the above steps is linearly dependent on the edges deleted thereby. Now we look at Reduction Rule 4.7. Here, we delete the degree-one node v. If u was unlabeled before applying Reduction Rule 4.7 on v and u has at least one more neighbor it likes exactly as much as v, then we add a label to u. As the preference lists are ordered, finding out if u has another node it likes as much as v has a constant running time. If u was already labeled before applying Reduction Rule 4.7, then we delete the label of u and add u to F. In this case we also delete all edges between u and the neighbors of u which it likes exactly as much as its label. Obviously, the running time of deleting edges is linearly dependent on the number of edges deleted. □

Lemma 4.14. *Reduction Rules 4.5 to 4.7 can be applied exhaustively in $\mathcal{O}(n + m)$ time.*

Proof. Let $(G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching. Searching for nodes v with $\deg(v) = 1$ takes $\mathcal{O}(n)$ time. Lemma 4.14 states that applying Reduction Rules 4.5 to 4.7 on a degree-one neighbor of a node $u \in V$ is

linear dependent on the edges deleted thereby. As each edge can be deleted at most once this leads to a running time of $\mathcal{O}(m)$ for applying Reduction Rules 4.5 to 4.7 on all known degree-one nodes.

By applying the data reduction rules new degree-one nodes could be produced. However, we can immediately check and remember if a new degree-one node was produced, when deleting the edge between u and a neighbor of u. The running time of this is linearly dependent on the number of edges deleted. Remembering the new degree-one nodes prevents us from checking for new degree-one nodes again. Thus, applying Reduction Rules 4.5 to 4.7 exhaustively has a running time of $\mathcal{O}(n + \sum_{u \in V(G)} \deg(u)) = \mathcal{O}(n + m)$. □

4.2 Paths

Next we want to develop a data reduction rule for paths, such that we can replace each long path by a path of constant length. Let $G = (V, E)$ be the input graph. Assume that G contains a subgraph Q whose inner nodes all have degree two in G. In less formal settings we will also refer to such subgraphs as paths. But, in the formal setting of data reduction rules we will always use the formally correct description. It is obvious that all inner nodes $v \in Q$ can only have first and second preferences, as they all have at most two neighbors. As we will later see the challenge in developing data reduction rules to confine the number of nodes in such a path is the potential labeling of nodes in it. If no node in the path is labeled, then the development of a data reduction rule is not as difficult. In this section, we will first present several data reduction rules that gradually remove all labels in paths. Afterwards, we can formulate the final reduction rule, that reduces all remaining paths to a constant length.

As already mentioned above, nodes in paths can only have first and second preferences. Even a labeled node $v \in V(Q)$ cannot have a third preference, as Reduction Rule 4.3 was already applied exhaustively. To limit the number of combinations of nodes with first and second preferences we can find in a path, we first apply Reduction Rule 4.4 exhaustively. Imagine now a path, with a node u, which has one first preference v and one second preference w. Node v must have u as its second preference, as otherwise the edge $\{u, w\}$ would have been deleted by Reduction Rule 4.4. The same has then to apply to v. Node v's first preference has to have v as its second preference. Thus, if nodes in a path contain second preferences, the path has to follow certain patterns, which are illustrated in Figure 4.7. Let Q be a path. We now fix the path such that all inner nodes in the path have a *left* neighbor and a *right* neighbor. If in a segment of a path all nodes have their right neighbor as second preference (and consequently their left neighbor as first preference), then we say that this segement follows the *1-2-pattern*. If in a segement of a path all nodes have their left neighbor as second preference, we say that the segment follows the *2-1-pattern*. Obviously, this pattern is the reverse of the *1-2-pattern*. If in a segment of a path all nodes have their left as well as their right neighbor as first preference, we say that the segment follows the *1-1-pattern*. Altogether, this leads to the following:

Observation 4.15. *After applying Reduction Rule 4.4 exhaustively, all subgraphs Q where the inner nodes have degree two in the input graph G, must follow the same order of*

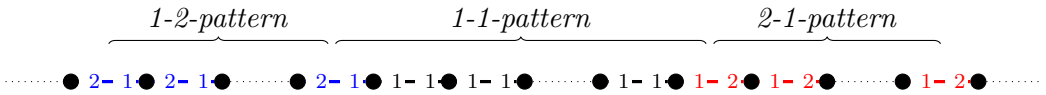


Figure 4.7: After applying Reduction Rule 4.4 exhaustively all paths have to follow certain patterns, as described in Observation 4.15. First, there can be a segment following the *1-2-pattern*, here represented in blue. Each node in this segment has its *left* neighbor as first preference and its *right* neighbor as second preference. This order of preferences can change into a segment following the *1-1-pattern*, where all nodes only have first preferences. Following this segment there can be one segment following the *2-1-pattern*, which is the reversion of the *1-2-pattern* concerning the order of preferences. This segment is represented in red. Each segment can contain an arbitrary number of nodes. Thus, a path does not have to contain all three patterns. However, if a path contains more than one segment, the different segments must follow the order described above. Hence, there cannot be a path that first has a segment following the *2-1-pattern*, followed by a segment following the *1-2-pattern*. If we had such a path, the connecting edge between these two segments would have been deleted by Reduction Rule 4.4. Thus, we would actually have two unconnected paths.

patterns: First we can have a segment following the 1-2-pattern*, attached to its right side a segment following the* 1-1-pattern*, and at the end a segment following the* 2-1-pattern*. Segments may be empty, but their order must be fixed in the above stated succession.*

We also note that while each of the three segments can have any length, there cannot be more than three segments in one path.

Looking at Figure 4.7 we can see that the *1-2-pattern* cannot be attached at the right of the *2-1-pattern*. The *1-1-pattern* can also neither be attached at the left of the *1-2-pattern*, nor at the right of the *2-1-pattern*. In all these cases Reduction Rule 4.4 would have deleted the edge which attaches the segments with such patterns.

We can view these three segments with different patterns as separate paths, which will make the formulation of data reduction rules easier. For the segment following the *1-1-pattern* we can apply a data reduction rule presented at the end of this section. Paths following the *1-2-pattern* and the *2-1-pattern* can be reduced using the same reduction rules, as the patterns are just a reversed version of each other. Thus, from here on forward we will only consider paths following the *1-2-pattern*. However, before formulating data reduction rules for such paths, we first have to look at the influence of the labeling of nodes on them. As in these paths all nodes have a second preference, it is possible to have labeled nodes in these paths. However, we can make the following observation.

Observation 4.16. *Let Q be a path following the* 1-2-pattern *and assume that Reduction Rule 4.2 is not applicable. If a node $v \in V(Q)$ is labeled, then the label is* 2*.*

Proof. Let Q be a path following the *1-2-pattern*. As Reduction Rule 4.2 is not applicable, nodes cannot be labeled with *1*. The last preferece of all inner nodes $v \in V(Q)$ is their second preference. Hence, nodes cannot be labeled with a number greater than *2*. Thus, the only possible label is *2*. □

When handling paths that contain labeled nodes p, we can distinguish two cases. In the first case no neighbors of p is labeled. In the second case one or both neighbor of p

are labeled. An important characteristic of the first case is described in the following lemma.

Lemma 4.17. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *and let Q be a subgraph of G such that the inner nodes of Q all have degree two in G and follow the* 1-2-pattern*. Furthermore let $p \in V(Q)$ be labeled and let p_{left} be the first preference of p and p_{right} be the second preference of p. Let nodes p_{left} and p_{right} be unlabeled. If nodes p and p_{right} are inner nodes of Q, then the edge $\{p, p_{\text{left}}\}$ must be contained in any annotated strongly stable matching in G.*

Proof. None of p's neighbors are labeled. Due to Observation 4.16 node p must be labeled with *2*. Thus, it cannot be matched to its second preference p_{right}. Node p either has to be matched to p_{left}, or p_{right} has to be matched to a neighbor which is not p, but which p_{right} likes at least as much as p. However, node p_{right} is also an inner node of the path following the *1-2-pattern*, and thus has its *left* neighbor p as first preference and its *right* neighbor as second preference. Hence, it is impossible for p_{right} to be matched to a neighbor it likes at least as much as p. Thus, we know that if p and p_{right} are inner nodes of the path, then the edge $\{p, p_{\text{left}}\}$ has to be part of the matching. □

In the data reduction rules for paths with the *1-2-pattern* we will see that only the inner nodes, which are not a neighbor of an outer node are deleted. Hence, if a labeled node p or one of its neighbors is an outer node, then node p does not get deleted in the reduction and we do not loose the information the label gives us.

Now we will examine the second case, where one or both neighbors of a labeled node in a path are labeled. Again, let Q be a subgraph of G such that the inner nodes of Q all have degree 2 in G and follow the *1-2-pattern*. Furthermore let $p \in V(Q)$ be labeled and let p_{left} be the first preference of p and p_{right} be the second preference of p. If only one neighbor of p is labeled, we only have to consider the case that p_{right} is labeled. If instead p_{left} was labeled, then we could view p_{left} as p and p as the right neighbor p_{right}. Again we only consider the case that p and its neighbors are inner nodes of the path, following the *1-2-pattern*. Because p is labeled, edge $\{p, p_{\text{right}}\}$ cannot be in the matching. Because p_{right} is labeled, node p_{right} cannot be matched to its own right neighbor. Thus, p_{right} cannot be matched at all. However, p_{right}'s right neighbor has p_{right} as single first preference if it is an inner node itself. As p_{right} is labeled with *2*, it is indifferent between being not matched and being matched to its right neighbor. Hence, the edge between p_{right} and its right neighbor is blocking if the right neighbor is an inner node of the path. Thus, if p_{right} is not neighbor to an outer node, then an annotated strongly stable matching does not exist. In the case that both of p's neighbors are labeled the same reasoning as above applies and an annotated strongly stable matching does not exist.

In general we can say that if we have a path $Q = (\{p_1, p_2, ..., p_\ell\}, \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\})$, and two nodes $p_x, p_y \in V(Q)$ are labeled, then the indices x and y must either be both even, or both odd for an annotated strongly stable matching to exist. There exists only one case in which an annotated strongly stable matching exists even though $x \bmod 2 \neq y \bmod 2$. That is, analogously to the second case in the consideration above, if one of p_x and p_y is the neighbor to the right outer node p_ℓ of the path Q. This case will be handled in the following data reduction rule, which is also visualized in Figure 4.8:

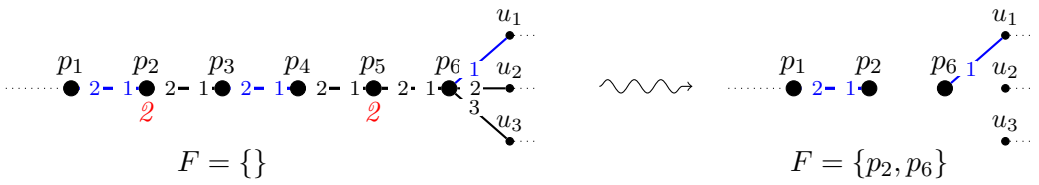


Figure 4.8: An example showing Reduction Rule 4.8. Nodes p_2 and p_5 are labeled. Because of its labeling p_2 cannot be matched to p_3. However, as the path follows the *1-2-pattern* node p_3 prefers p_2 to its other neighbor p_4. Thus, node p_2 has to be matched to a node it prefers to p_2 and its label. Hence, we add p_2 to F. In this example p_2 has to be matched to p_1. As p_4 has p_3 as its first preference and p_3 cannot be matched to its own first preference, edge $\{p_3, p_4\}$ must be in the matching. Thus, p_5 cannot be matched. As we know which edges between p_2 and p_6 have to be in the matching, we can delete this segment of the path. For the edge $\{p_5, p_6\}$ not to be blocking, node p_6 must be matched to a node it likes at least as much as p_5, which is only u_1 in this example. Thus, delete edges $\{p_6, u_2\}$ and $\{p_6, u_3\}$ and add p_6 to F.

Reduction Rule 4.8. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching. *Let Q be a subgraph of G with $V(Q) = \{p_1, p_2, ..., p_\ell\}$ and $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\}$ such that the inner nodes of Q all have degree 2 in G and follow the* 1-2-pattern. *Let $p_x, p_{\ell-1} \in V(Q)$ be labeled with 2 and $x \in \{1, ..., \ell - 2\}$ such that $x \bmod 2 \neq (\ell - 1) \bmod 2$. Moreover, no labeled node $p_i \in V(Q)$, with $x < i < \ell - 1$ exists.*

Delete the label of p_x, add p_x and p_ℓ to F, delete each edge $\{p_\ell, v\}$, where $p_{\ell-1} \prec_{p_\ell} v$ and delete all p_i with $x < i \leq \ell - 1$. Moreover, we delete all edges $\{p_x, v\}$ where v is not a first preference of p_x.

Lemma 4.18. *Reduction Rule 4.8 is correct.*

Proof. Let G be the input graph and Q a subgraph of G such that the inner nodes of Q all have degree 2 in G and follow the *1-2-pattern*. Let $V(Q) = \{p_1, p_2, ..., p_\ell\}$ and $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\}$. Let $p_x \in V(Q)$ be a labeled node of Q with a second preference p_{x+1}. Furthermore, let $F(G)$ be the set of nodes that have to be matched in G. Let G' be the reduced graph, where $F(G')$ is the set of nodes that have to be matched.

Node $p_{\ell-1}$ is an inner labeled node of Q with second preference p_ℓ. As we apply Reduction Rule 4.8 nodes p_x and $p_{\ell-1}$ are labeled with *2*.

First, given an annotated strongly stable matching M in G, we construct an annotated strongly stable matching M' in G'. As both p_x and $p_{\ell-1}$ are labeled with *2* the edges $\{p_x, p_{x+1}\}$ and $\{p_{\ell-1}, p_\ell\}$ cannot be in the matching M. We will now show that $p_{\ell-1}$ cannot be matched to its first preference $p_{\ell-2}$ either. As p_x cannot be matched to p_{x+1}, node p_{x+1} can either be unmatched or matched to its own second preference and right neighbor p_{x+2}. Obviously, node p_{x+1} prefers being matched to p_{x+2} to being unmatched. Moreover, as the path follows the *1-2-pattern*, node p_{x+2} prefers p_{x+1} to its own second preference and right neighbor. Hence, the edge $\{p_{x+1}, p_{x+2}\}$ must be in the matching and thus the edge between p_{x+2} and p_{x+2}'s own second preference cannot be in the matching. If we now look at the second preference of p_{x+2} and the second preference of that node, we see that those nodes are now in the same situation as p_{x+1}

and p_{x+2} before. As no node $p_i \in V(Q)$, with $x < i < \ell - 1$ is labeled, we can continue looking at the nodes in the path in the same fashion, until we reach $p_{\ell-1}$, such that beginning with the edge $\{p_{x+1}, p_{x+2}\}$ every second edge is in the matching M. Thus, as $x \bmod 2 \neq (\ell-1) \bmod 2$ the edge $\{p_{\ell-3}, p_{\ell-2}\}$ is in the matching M and the node $p_{\ell-1}$ cannot be matched in M. The edge $\{p_{\ell-2}, p_{\ell-1}\}$ is not blocking as $p_{\ell-2}$ prefers $p_{\ell-3}$ to $p_{\ell-1}$. As $p_{\ell-1}$ is labeled, it is indifferent between not being matched and being matched to p_ℓ. For the edge $\{p_{\ell-1}, p_\ell\}$ not to be blocking, node p_ℓ must be matched to a node it likes at least as much as $p_{\ell-1}$. In the reduced graph G', node p_ℓ must also be matched, because $p_\ell \in F(G')$. As in G' node p_ℓ only has neighbors it likes at least as much as it liked $p_{\ell-1}$ in G, it must be matched to one of them. This is already satisfied by matching M.

Next we look at the node p_x. First we look at the case, that p_x is an inner node of Q. In this case we already established that p_x cannot be matched to p_{x+1}. Let p_{x-1} be the first preference of p_x. As Lemma 4.17 states, for the edge $\{p_x, p_{x+1}\}$ not to be blocking, node p_x must be matched to p_{x-1}, the only node p_x prefers to p_{x+1}. Thus, it is sufficient for p_x to be an element of $F(G')$. Now we look at the case that p_x is an outer node of Q, i.e., $x = 1$. In this case node p_x also has to be matched to a first preference so that $\{p_x, p_{x+1}\}$ is not blocking. We force this by deleting all edges to nodes that p_x likes less than p_{x+1} and adding p_x to $F(G')$. Hence $M \setminus \{\{p_i, p_{i+1}\} \mid x < i < y\}$ is an annotated strongly stable matching in G'.

Let M' be an annotated strongly stable in G'. Node p_ℓ must be matched in M' as $p_\ell \in F(G')$. In the graph G, node p_ℓ must be matched to a node it likes at least as much as $p_{\ell-1}$. This is also satisfied in M', as in the reduced graph G' node p_ℓ only has neighbors it likes at least as much as $p_{\ell-1}$ in G. Furthermore, as we already established in the other direction of the proof, in a matching in G node p_x must be matched to a node it prefers to its second preference p_{x+1}. This is also satisfied in M', as $p_x \in F(G')$ and p_x only has first preferences in G'. In the reduction we also delete all edges $\{\{p_i, p_{i+1}\} \mid x < i < y\}$. We know that every second of those edges must be in a matching in G. Thus, the matching $M' \cup \{\{p_i, p_{i+1}\} \mid x < i < y\}$ is an annotated strongly stable matching in G. □

After applying Reduction Rule 4.8 exhaustively, we know that if for any two labeled nodes in a path $Q = (\{p_1, p_2, ..., p_\ell\}, \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\})$ their indices are not both even or both odd, then an annotated strongly stable matching does not exist. The only exception to this is the case in which the last node p_ℓ in the path is labeled. We did not consider this case in Reduction Rule 4.8. However, this does not limit possible matchings within the path Q at all. Imagine the node $p_\ell \in V(Q)$ would be labeled with *2*. This would mean that p_ℓ could no be matched to a second preference. But the only incident edge node p_ℓ has within the path Q is connected to its first preference $p_{\ell-1}$. Whether p_ℓ can be matched to $p_{\ell-1}$ or not is not influenced by its label. Due to the same reason a higher label would also not influence our data reduction rules concerning paths.

The next data reduction rule will consider all remaining paths that contain two labeled nodes that do not have both even ore both indices. Apart from the case where one of those nodes is the last node in the path we can immediately stop our algorithm. An example for this is visualized in Figure 4.9.

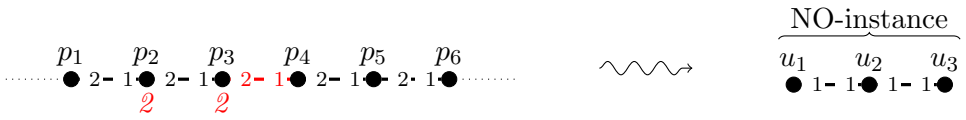


Figure 4.9: An example showing Reduction Rule 4.9. Nodes p_2 and p_3 are labeled. Thus neither edge $\{p_2, p_3\}$ nor $\{p_3, p_4\}$ can be in the matching. Thus, node p_3 cannot be matched. However, as the path follows the *1-2-pattern*, node p_4 prefers p_3 to its only other neighbor p_5. Hence, edge $\{p_3, p_4\}$ is blocking and an annotated strongly stable matching does not exist. We can replace the whole graph by a simple NO-instance.

Reduction Rule 4.9. *Let* $(G = (V, E), F, L, P)$ *be an instance of* Annotated Strongly Stable Matching *where Reduction Rule 4.8 is not applicable. Let* Q *be a subgraph of* G *with* $V(Q) = \{p_1, p_2, ..., p_\ell\}$ *and* $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\}$ *such that the inner nodes of* Q *all have degree 2 in* G *and follow the* 1-2-pattern*. Let* $p_x, p_y \in V(Q)$ *with indices* $x, y \in \{1, 2, ..., \ell - 2\}$ *be labeled with* 2*. If* $x \mod 2 \neq y \mod 2$*, then replace* G *by a simple NO-instance.*

Lemma 4.19. *Reduction Rule 4.9 is correct.*

Proof. Let G be a graph where Reduction Rule 4.8 is not applicable and Q a subgraph of G such that the inner nodes of Q all have degree 2 in G and follow the *1-2-pattern*. Let $V(Q) = \{p_1, p_2, ..., p_\ell\}$ and $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ...\ell - 1\}\}$.

We will now show that if G contains a strongly stable matching M, then every node $p_i \in V(Q) \setminus \{p_{ell-1}, p_\ell\}$ must be matched in it. Let $p_{i+1} \in V(Q)$ be the second preference of node $p_i \in V(Q)$, where $i \in \{1, 2, ..., \ell - 2\}$. As Q follows the *1-2-pattern*, node p_i has to be the only first preference of node p_{i+1}. Thus, if p_i is not matched, then edge $\{p_i, p_{i+1}\}$ would be blocking. Again, the only exception here is caused by node p_ℓ, as $p_{\ell-1}$ does not have to be the only first preference of p_ℓ.

Let $p_y \in V(Q)$ be labeled with *2* and $y \in \{2, ..., \ell - 2\}$. Node p_y is an inner node of Q. Thus, as stated in Lemma 4.17 node p_y must be matched to its first preference p_{y-1}, i.e., $\{p_{y-1}, p_y\} \in M$.

As every node in $V(Q)$ but $v_{\ell-1}$ has to be matched, every second edge in the path has to be in the matching. We know that $\{p_{y-1}, p_y\}$ must be in the matching M. This clearly determines which edges $\{p_i, p_{i+1}\}$ with $1 \leq i < y$ have to be in the matching M. Let node $p_x \in V(Q)$ be another labeled node, with $1 \leq x < y$ and $x \mod 2 \neq y \mod 2$. Node p_x is also labeled with *2*. Thus, p_x cannot be matched to its second preference p_{x+1}. As the path follows the *1-2-pattern* we know that p_{x+1} has p_x as its only first preference. We also know that p_{x+1}'s second preference p_{x+2} has p_{x+1} as only first preference. As p_{x+1} prefers being matched to its second preference to not being matched, we can follow that edge $\{p_{x+1}, p_{x+2}\}$ has to be in the matching. However, as $x \mod 2 \neq y \mod 2$ this edge is not in the set of edges for which we previously determined that they have to be in the matching. Thus, either the edge $\{p_{x+1}, p_{x+2}\}$ or the edge $\{p_{y-1}, p_y\}$ is blocking. Hence, an annotated strongly stable matching does not exist and we can replace the graph G by a simple NO-instance. □

After also applying Reduction Rule 4.9 exhaustively, we know that no two labeled

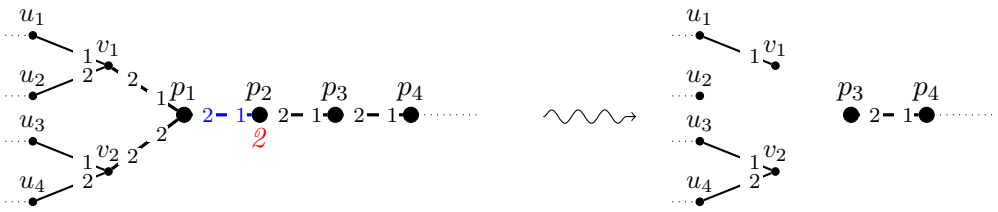


Figure 4.10: An example showing Reduction Rule 4.10. Node p_2 is labeled. Thus, it cannot be matched to its second preference p_3. However, as the path follows the *1-2-pattern*, p_2 must be matched to p_1 such that edge $\{p_2, p_3\}$ is not blocking. Thus, we delete nodes p_1 and p_2. But we also have to consider that node p_1 may prefer another node to p_2. Node p_1 has v_1 as its first preference and consequently prefers v_1 to p_2. Thus, in order for edge $\{p_1, v_1\}$ not to be blocking, node v_1 must be matched to a node it prefers to p_1. Hence, we add v_1 to F and delete the edges between v_1 and neighbors of v_1 which it likes less or the same as p_1. Thus, only edge $\{v_1, u_1\}$ remains. Node v_2 is a second preference of p_1. Thus, p_1 is indifferent between v_2 and p_2. For edge $\{p_1, v_2\}$ not to be blocking, node v_2 must be matched to a node it likes at least as much as p_1. Hence, we add v_2 to F and delete all edges between v_2 and neighbors of v_2 which it likes less than p_1.

nodes in a path can be neighbors (with the exception of the right outer node and its neighbor). We can now apply the following data reduction rule, which is illustrated in Figure 4.10, to delete any still remaining labels in paths.

Reduction Rule 4.10. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *where Reduction Rule 4.9 is not applicable. Let $Q = (\{p_1, p_2, ..., p_\ell\}, \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\})$ be a subgraph of G such that the inner nodes of Q all have degree 2 in G and follow the* 1-2-pattern. *Let $p \in V(Q) \setminus \{p_1, p_{\ell-1}, p_\ell\}$ be labeled, node p_{left} be p's first preference and node p_{right} be p's second preference. Nodes p and p_{right} are inner nodes of Q.*

Delete node p and p_{left}. For each node $v \in N_G(p_{\text{left}}) \setminus \{p\}$, add v to $F(G)$. Furthermore,

(a) if p_{left} is indifferent between p and v, then delete the edges between v and all neighbors of v, which v likes less than p_{left} and

(b) if p_{left} prefers v to p, then delete all edges from v to neighbors of v, which node v likes less or the same than p_{left}.

Lemma 4.20. *Reduction Rule 4.10 is correct.*

Proof. Let G be a graph where Reduction Rule 4.9 is not applicable and Q be a subgraph of G such that the inner nodes of Q all have degree 2 in G and follow the *1-2-pattern*. Let $V(Q) = \{p_1, p_2, ..., p_\ell\}$ such that edges $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\}$. Let $p \in V(Q) \setminus \{p_1, p_{\ell-1}, p_\ell\}$ be a labeled inner node of Q. Because of Observation 4.16, node p must be labeled with *2*. Let p_{left} be p's first preference and let p_{right} be p's second preference and an inner node of Q. As Reduction Rule 4.9 is not applicable, node p_{left} and node p_{right} cannot be labeled. Let M be an annotated strongly stable matching in G and let $F(G)$ be the set of nodes that have to be matched in M. Furthermore, let $F(G')$

be the set of nodes that have to be matched in an annotated strongly stable matching in the reduced graph G'. As Lemma 4.17 states, edge $\{p, p_{\text{left}}\}$ must be in the matching M. Then, node p_{left} cannot be matched to any of its other neighbors $v \in N_G(p_{\text{left}}) \setminus \{p\}$. As Reduction Rule 4.4 is not applicable, node p_{left} must like each neighbor $v \in N_G(p_{\text{left}}) \setminus \{p\}$ at least as much as p. Hence, p is either indifferent between v and p or prefers v to p. If p_{left} is indifferent between node v and p, then for the edge $\{p_{\text{left}}, v\}$ not to be blocking, node v must be matched to a node it likes at least as much as p_{left} in the matching M. In this case, during the reduction all neighbors of v which it likes less than p_{left} are deleted and node v cannot be matched to these nodes in the graph G'. If p_{left} prefers node v to node p, then for the edge $\{p_{\text{left}}, v\}$ not to be blocking, node v must be matched to a node it prefers to p_{left} in the matching M. In this case, during the reduction all neighbors of v which it likes less or the same as p_{left} are deleted and node v cannot be matched to these nodes in the graph G'.

During the reduction each $v \in N_G(p_{\text{left}}) \setminus \{p\}$ is added to the set $F(G')$. Thus, such a node v must also be matched in an annotated strongly stable matching in G'. As p_{left} is deleted during the reduction v must be matched to a node $u \in N_G(v) \setminus \{p_{\text{left}}\}$ in the matching in G'. Therefore, the matching $M \setminus \{\{p, p_{\text{left}}\}\}$ is annotated strongly stable in G'.

In an annotated strongly stable matching M' in the reduced graph G' all nodes $v \in N_G(p_{\text{left}})$ are matched as for each v it applies that $v \in F(G')$. They must be matched to nodes they either likes at least as much as p_{left} in G, or prefer to p_{left} in G, depending on their preference for p_{left} in G. The reason for this is that in G' they only have neighbors that satisfy this requirement. As we have already established in the other direction of the proof, the nodes $v \in N_G(p_{\text{left}})$ have to be matched the same way in an annotated strongly stable matching in G, too. In M' no node v is matched to p_{left}, as $p_{\text{left}} \notin V(G')$. For node p it also applies that $p \notin V(G')$. Node p and node p_{left} are the only nodes that exist in G, but not in G'. Moreover, node p and node p_{left} must be matched to each other in G. Hence, $M' \cup \{\{p, p_{\text{left}}\}\}$ is annotated strongly stable in G. □

After having applied Reduction Rule 4.10 paths do not contain labeled inner nodes anymore. The only exception to this is the left neighbor of the right outer node. This node could still be labeled as we excluded this case in Reduction Rule 4.10. The reason for this is that in this case Lemma 4.17 does not apply. While this possible labeling does influence the matching, it does not influence the following data reduction rule. This is our final data reduction rule that reduces all remaining paths to a constant length. However, the outer nodes as well as their neighbors are not deleted during this data reduction rule and therefore any possible labels of those nodes remain. The idea behind this data reduction rule is also illustrated in Figure 4.11.

Reduction Rule 4.11. *Let* $(G = (V, E), F, L, P)$ *be an instance of* Annotated Strongly Stable Matching *where Reduction Rule 4.10 is not applicable. Let* Q *be a subgraph of* G *with* $V(Q) = \{p_1, p_2, ..., p_\ell\}$ *with* $\ell > 5$ *and* $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\}$ *such that the inner nodes of* Q *follow either the* 1-2-pattern *or the* 1-1-pattern. *All inner nodes of* Q *have degree 2 in* G.

(a) If the number of nodes in Q *is even, then delete the inner nodes* $\{p_i \mid i \in \{3, ..., \ell-2\}\}$ *and add the edge* $\{p_2, p_{\ell-1}\}$.

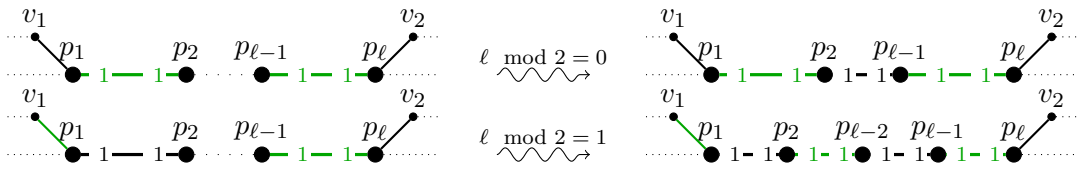


Figure 4.11: Examples showing Reduction Rule 4.11. The path must follow the *1-1-pattern* or *1-2-pattern* and no inner node contained in the path can be labeled. Thus, if an annotated strongly stable matching exists, then every second edge in the path must be in the matching. Either the annotated strongly stable matching of the input graph and the strongly stable matching of the reduced graph both contain the green edges, or they both contain the black edges. In the case of an even number of nodes in the path, either edges $\{p_1, p_2\}$ and $\{p_{\ell-1}, p_\ell\}$ are both in the matching or both not in the matching. In the case of an odd number of nodes in the path, exactly one of the edges $\{p_1, p_2\}$ and $\{p_{\ell-1}, p_\ell\}$ is in the matching. To retain this requirement we replace a path with an even number of nodes by a shorter path with an even number of nodes and we replace a path with an odd number of nodes by a shorter path with an odd number of nodes.

(b) *If the number of nodes in Q is odd, then delete the inner nodes $\{p_i \mid i \in \{3, ..., \ell-3\}\}$ and add the edge $\{p_2, p_{\ell-2}\}$.*

The pattern in the reduced path follows the same pattern of preferences as the path Q.

Lemma 4.21. *Reduction Rule 4.11 is correct.*

Proof. Let $(G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching and Q a subgraph of G such that the inner nodes of Q all have degree 2 in G. Let $V(Q) = \{p_1, p_2, ..., p_\ell\}$ with edges $E(Q) = \{\{p_i, p_{i+1}\} \mid i \in \{1, ..., \ell - 1\}\}$.

We will now show that if G contains a strongly stable matching M, then every node $p_i \in V(Q) \setminus \{p_{\ell-1}, p_\ell\}$ must be matched in it. Looking at paths following the *1-2-pattern*, let $p_{i+1} \in V(Q)$ be the second preference of node p_i. Then p_i has to be the first preference of node p_{i+1}. If p_i is not matched, then the edge $\{p_i, p_{i+1}\}$ would be blocking, as p_i prefers p_{i+1} to not being matched, and p_{i+1} either prefers p_i to its other neighbor, or is indifferent between both its neighbors. In paths following the *1-1-pattern*, if a node $p_i \in V(Q)$ is not matched, then the edge $\{p_i, p_{i+1}\}$ would be blocking, as p_i prefers p_{i+1} to not being matched, and p_{i+1} is indifferent between both its neighbors. This especially means, that every second edge contained in Q has to be part of the matching. The only exception to this is the edge $\{p_{\ell-1}, p_\ell\}$ in case $p_{\ell-1}$ is labeled.

First we consider the case where $p_{\ell-1}$ is not labeled. Let an even path Q have an even number of nodes $V(Q)$ and an odd path Q an odd number of nodes $V(Q)$. If Q is even, then either both p_1 and p_ℓ are matched to their neighbors p_2 and $p_{\ell-1}$, which are contained in the path Q, or both are matched to neighbors which are not contained in Q. If Q is odd, then one of p_1 and p_ℓ has to be matched to its neighbor which is contained in the path and the other one has to be matched to a neighbor which is not contained in the path.

As Reduction Rule 4.10 is not applicable and $p_{\ell-1}$ is not labeled, the path cannot contain labeled nodes. First we look at the case of even paths. Let G' be the reduced graph, where Q was replaced by $Q' = (\{p_1, p_2, p_{\ell-1}, p_\ell\}, \{\{p_1, p_2\}, \{p_2, p_{\ell-1}\}, \{p_{\ell-1}, p_\ell\}\})$. If the matching M in G contains the edges $\{p_1, p_2\}$ and $\{p_{\ell-1}, p_\ell\}$, then $M \setminus \{\{p_i, p_{i+1}\} \mid$

$i \in \{3, 5, \ell - 3\}\}$ is an annotated strongly stable matching in G'. If the matching M in G contains the edges $\{v_1, p_1\}$ and $\{p_\ell, v_2\}$, where $v_1, v_2 \in V(G) \backslash V(Q)$, then $(M \backslash \{\{p_i, p_{i+1}\} \mid i \in \{2, 4, 6, ..., \ell - 2\}\}) \cup \{\{p_2, p_{\ell-1}\}\}$ is an annotated strongly stable matching in G'.

If an annotated strongly stable matching M' in G' exists, and $\{p_1, p_2\}, \{p_{\ell-1}, p_\ell\} \in M'$, then $(M' \cup \{\{p_i, p_{i+1}\} \mid i \in \{3, 5..., \ell - 3\}\}$ is an annotated strongly stable matching in G. If the matching M' contains the edges $\{v_1, p_1\}$ and $\{p_\ell, v_2\}$, where $v_1, v_2 \in V(G') \setminus V(Q')$, then $(M' \setminus \{\{p_2, p_{\ell-1}\}\}) \cup \{\{p_i, p_{i+1}\} \mid i \in \{2, 4, 6, ..., \ell - 2\}\}$ is an annotated strongly stable matching in G.

In the case of odd paths let G' be the reduced graph, where Q was replaced by $Q' = (\{p_1, p_2, p_{\ell-2}, p_{\ell-1}, p_\ell\}, \{\{p_1, p_2\}, \{p_2, p_{\ell-2}\}, \{p_{\ell-2}, p_{\ell-1}\}, \{p_{\ell-1}, p_\ell\}\})$. Without loss of generality we can say that if a strongly stable matching M exists in G, then the edges $\{v, p_1\} \in M$ and $\{p_{\ell-1}, p_\ell\} \in M$, where $v \in V(G) \setminus V(Q)$, are in the matching. Then $(M \setminus \{\{p_i, p_{i+1}\} \mid i \in \{2, 4, 6, ..., \ell - 3\}\}) \cup \{\{\{p_2, p_{\ell-2}\}\}$ is an annotated strongly stable matching in G'.

If an annotated strongly stable matching M' in G' exists, then without loss of generality we can say that $\{v, p_1\} \in M'$ and $\{p_2, p_{\ell-2}\} \in M'$, where $v \in V(G') \setminus V(Q')$, are contained in the matching. Then $(M' \setminus \{\{p_2, p_{\ell-2}\}\}) \cup \{\{p_i, p_{i+1}\} \mid i \in \{2, 4..., \ell - 1\}\}$ is an annotated strongly stable matching in G.

We will now consider the case that $p_{\ell-1}$ is labeled. In this case the path Q must follow the *1-2-pattern*. In a path following the *1-1-pattern* the existence of labels is not possible. As stated in Observation 4.16, node $p_{\ell-1}$ must be labeled with *2*. Thus, it cannot be matched to its only second preference p_ℓ. Now, there are two possible cases. Either the edge $\{p_{\ell-2}, p_{\ell-1}\}$ is in the matching, or not.

First we look at the case that $\{p_{\ell-2}, p_{\ell-1}\}$ is in the matching. In this case $\{p_{\ell-1}, p_\ell\}$ is not blocking, independent of p_ℓ's matching partner. Starting from the edge $\{p_{\ell-2}, p_{\ell-1}\}$ and going towards p_1 in the path, every second edge in the path must be in the matching. More specifically, the edges $\{\{p_{\ell-i}, p_{\ell-(i+1)}\} \mid i \in \{1, 3, 5, ...\} \wedge (\ell - i) > 1\}\}$ have to be in the matching. This applies for both the graph G and the reduced graph G'. Thus, as in both graphs the path Q and Q' are either both odd, or both even an annotated strongly stable matching in G exists if and only if an annotated strongly stable matching in G' exists. The exact transformation of the matching in G to the matching in G' and vice versa can be done accordingly to the transformations described above.

Now we look at the case that edge $\{p_{\ell-2}, p_{\ell-1}\}$ is not in the matching. Again every second edge on the path between p_1 and $p_{\ell-1}$ must be in the matching, both in the path Q and the path Q'. However, as $\{p_{\ell-2}, p_{\ell-1}\}$ is not in the matching this is the set of edges that were not in the matching before as we looked at the other case. More specifically the edges $\{\{p_{\ell-i}, p_{\ell-(i+1)}\} \mid i \in \{2, 4, 6, ...\} \wedge (\ell - i) > 1\}\}$ have to be in the matching. Now, for the edge $\{p_{\ell-1}, p_\ell\}$ not to be blocking, node p_ℓ has to be matched to a node it likes at least as much as $p_{\ell-1}$, i.e., another first preference. This also applies for both graphs G and G'. Thus, in this case an annotated strongly stable matching exists in G if and only if one exists in G'. □

Now we have developed all data reduction rules which we use to reduce paths to a constant length. If we take a closer look at the data redcution rules, we can see that by applying Reduction Rules 4.8 and 4.10 new degree-one nodes are produced. Thus, whenever Reduction Rules 4.8 and 4.10 are applied on a path, we can delete the whole

path by repeatedly applying the degree-one data reduction rules afterwards. In Reduction Rule 4.9 we also have no path remaining, as we return a NO-instance. The only data reduction rule where we still have a path after application is Reduction Rule 4.11. If the path we applied Reduction Rule 4.11 on had an even number of nodes, then we replaced it with a path of four nodes. If the path had an odd number of nodes, then we replaced it with five nodes. Recall that for the data reduction rules we divided paths into subpaths such that each subpath followed one pattern. The data reduction rules are applied seperately on each subpath. Thus, after we applied Reduction Rule 4.11 on all subpaths each subpath can have a maximum of five nodes. In Observation 4.15 we stated that each path can consist of a maximum of three subpaths. We also have to take into account that the inner subpath shares its outer nodes with the other two subpaths. Altogether this leads to the following observation.

Observation 4.22. *Let $(G = (V, E), F, L, P)$ be an instance of* Annotated Strongly Stable Matching *where Reduction Rules 4.8 to 4.11 are not applicable. Let Q be a path in G such that the inner nodes of Q have degree two in G. Then Q contains at most 13 nodes of which 11 nodes are inner nodes.*

We conclude this section by showing that Reduction Rules 4.8 to 4.11 can be executed in linear time. In Reduction Rule 4.8 we delete nodes of a path such that new degree-one nodes are produced. We can then delete all remaining nodes of this path by degree-one data reduction rules. Imagine the following scenario. There are three paths with one common outer node. If we apply Reduction Rule 4.8 on one path and afterwards delete the rest of this path, then we are left with two paths with one common outer node. If these two paths follow a common pattern, they might actually be one path and we have to join them. The running time of joining two paths brute-force is linearly dependent on the length of the paths. This poses a problem as we might have to join two paths each time we apply Reduction Rule 4.8 on a path. In total this would lead to a worst-case complexity of $\mathcal{O}(n^2)$. To reach a worst-case complexity of $\mathcal{O}(n)$ we need to store the paths in a data structure that allows for more efficient joining.

Lemma 4.23. *Let $G = (V, E)$ be a graph. Let Q_1 and Q_2 be two subgraphs of G with $V(Q_1) = \{p_1, p_2, ..., p_{i-1}, p_i\}, E(Q_1) = \{\{p_j, p_{j+1}\} \mid j \in \{1, ..., i-1\}\}, V(Q_2) = \{p_i, p_{i+1}, ..., p_\ell\}$ and $E(Q_2) = \{\{p_j, p_{j+1}\} \mid j \in \{i, ..., \ell - 1\}\}$. All inner nodes of Q_1 and Q_2 have degree 2 in G. For each path we store the first, second, second-last and last node of the path, as well as the pattern the path follows.*

If (a) paths Q_1 and Q_2 share one common outer node p_i, which has no neighbors besides $p_{i-1} \in V(Q_1)$ and $p_{i+1} \in V(Q_2)$ and (b)

(b1) *Q_1 and Q_2 both follow the* 1-1-pattern*, or*

(b2) *Q_1 and Q_2 follow the same pattern and p_i is the last node of one of the paths and the first node of the other path, or*

(b3) *Q_1 follows the* 1-2-pattern *and Q_2 follows the* 2-1-pattern *and p_i is the last node of Q_1 and Q_2,*

then Q_1 and Q_2 can be joined to become one single path $Q_{1\cdot2}$ with $V(Q_{1\cdot2}) = \{p_1, p_2, ..., p_\ell\}$ and $E(Q_{1\cdot2}) = \{\{p_j, p_{j+1}\} \mid j \in \{1, ..., \ell - 1\}\}$ in constant time.

Proof. Let $G = (V, E)$ be a graph. Let Q_1 and Q_2 be two subgraphs of G with $V(Q_1) = \{p_1, p_2, ..., p_{i-1}, p_i\}, E(Q_1) = \{\{p_j, p_{j+1}\} \mid j \in \{1, ..., j-1\}\}, V(Q_2) = \{p_i, p_{i+1}, ..., p_\ell\}$ and $E(Q_2) = \{\{p_j, p_{j+1}\} \mid j \in \{i, ..., \ell\}\}$.

W.l.o.g we say that p_1 is the first node of Q_1, p_2 is the second node of Q_1, p_{i-1} is the second last node of Q_1 and p_i is the last node of Q_1.

For us to be able to join Q_1 and Q_2 to one single path $Q_{1\cdot2}$ the paths must satisfy certain conditions. We will now prove that with these conditions and the described data structure a joining in constant time is possible.

Point (a) in Lemma 4.23 ensures that all inner nodes of the joined path $Q_{1\cdot2}$ have degree two in G.

In order for $Q_{1\cdot2}$ to follow one single pattern, paths Q_1 and Q_2 must follow certain preference patterns. Recall that to determine a pattern of a path we gave the path a direction, such that each inner node contained in the path had a distinct left and right neighbor. For this data structure we say, that the left neighbor of each inner node is the neighbor of the node that has the shorter distance to the node which is stored as first node of the path. Consequently, the right neighbor of each node is the neighbor of the node that has the shorter distance to the node which is stored as last node of the path. If Q_1 and Q_2 both follow the *1-1-pattern*, then $Q_{1\cdot2}$ follows the *1-1-pattern*.

We already determined that p_i is the last node of Q_1. Thus, if Q_1 follows the *1-2-pattern*, then each node in $V(Q_1)$ has the node that has a smaller distance to p_i as second preference. If Q_2 also follows the *1-2-pattern*, then p_i must be the first node of Q_2. Otherwise, the nodes in $V(Q_2)$ would also have the neighbor as second preference which has a smaller distance to p_i. Thus, if we wanted to join Q_1 and Q_2 the joined path would change its pattern at p_i which we do not want. For the same reason, node p_i must also be the last node of Q_2, if Q_2 follows the *2-1-pattern*. The same also applies if Q_1 follows the *2-1-pattern*. Then Q_2 must also follow the *2-1-pattern* and p_i must be the first node of Q_2. However it is not possible for Q_1 to follow the *2-1-pattern* and Q_2 to follow the *1-2-pattern*, as shown in Observation 4.15.

We form path $Q_{1\cdot2}$ by storing p_1 as first node of $Q_{1\cdot2}$, p_2 as second node of $Q_{1\cdot2}$, $p_{\ell-1}$ as second last node of $Q_{1\cdot2}$ and p_ℓ as last node of $Q_{1\cdot2}$. As all four nodes were already stored in the data structures of Q_1 and Q_2 we do not have to search for them and adding them to the data structure of $Q_{1\cdot2}$ takes constant time.

Then we look at p_2, the second node of $Q_{1\cdot2}$. If p_2 has two first preferences, then we store that $Q_{1\cdot2}$ follows the *1-1-pattern*. If p_1 is the first preference of p_2 and p_2's other neighbor is the second preference of p_2, then we store that $Q_{1\cdot2}$ follow the *1-2-pattern*. If p_1 is the second preference of p_2 and p_2's other neighbor is the first preference of p_2, then we store that $Q_{1\cdot2}$ follow the *2-1-pattern*. Hence, for detecting the pattern of $Q_{1\cdot2}$ we only have to look up two preferences, which can be done in constant time.

Furthermore, we need to delete the data structures for paths Q_1 and Q_2. For each path we store just five values. Thus, deleting the paths can also be done in constant time.

Altogether this leads to a constant time for joining two paths. □

Using Lemma 4.23 we can now prove the linear running time of the data reduction rules presented in this section.

Lemma 4.24. *Reduction Rules 4.8 to 4.11 can be applied exhaustively in $\mathcal{O}(n+m)$ time.*

Proof. Let $(G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching. Before applying Reduction Rules 4.8 to 4.11, we have to find all subgraphs Q of G where the inner nodes of Q have degree two in G. Therefor we can start at any node $v \in V$. If $\deg(v) = 2$, then we continue to look at the neighbors of v. If they only have one neighbor besides v we continue to look at those neighbors and their neighbors again and so on. When we reach a node $u \in V$ that has more than two neighbors, then we know that we have reached the end of the path. We have also reached the end of a path if we reach a node that does not follow the same preference pattern as the nodes already contained in the path, as we view sections of a path with different preference patterns as separate paths. We are only interested in paths that contain more than five nodes. If we find a node that is not element of a path, then we remember it, such that we will not have to consider it again in the search for paths. If we find a path Q with a length greater than five, then we remember for each path the first, second, second-last and last node, as well as the pattern it follows. We apply the data reduction rules in the following order:

(1) First we apply Reduction Rule 4.8 exhaustively on all paths. Whenever we apply Reduction Rule 4.8 on a path Q we have to delete two labels, which takes constant time. We also delete at most all inner nodes of the path, as well as all neighbors of the right outer node of the path. The running time of this is linearly dependent on the length of the path Q and the degree of the right outer node of Q. Afterwards, we are potentially left with two more degree-one vertices, which we remember. Then, we apply the degree-one data reduction rules on them and all degree-one nodes that arise thereof. The degree-one node on which the degree-one data reduction rule is applied is deleted during the reduction. Hence, a degree-one data reduction rule can be applied on a maximum of all nodes in the graph. Moreover, the running time of applying degree-one data reduction rules on a degree-one neighbor of a node is linearly dependent on the number of edges deleted thereby, as proven in Lemma 4.13. Therefore, applying degree-one data reduction rules each time after having applied the Reduction Rule 4.8 takes $\mathcal{O}(m)$ time, as each edge in a graph can be deleted at most once. After we applied the degree-one data reduction rules on all new degree-one nodes, we have deleted all inner nodes of the path Q. If after the application of the Reduction Rule 4.8 and degree-one data reduction rules the left outer node of Q or the right outer node of Q have degree two, then we might have to join two paths in which this node is contained. For the following we will call this degree-two node p. If p is contained in two paths which satisfy the conditions stated in Lemma 4.23 then we join those paths which can be done in constant time (see Lemma 4.23). If p is contained in one path and p has one neighbor p' which is not contained in that path, but follows the same pattern as the path, then we add this neighbor to the path. To do so we change the first or respectively last node stored for the path form p to p' and change the second or respectively second-last node of the path to p. As this takes only four steps, it can be done in constant time. Altogether, this leads to a running time of $\mathcal{O}(n + m)$ for applying Reduction Rule 4.8 exhaustively on all paths.

(2) Next we apply Reduction Rule 4.9 exhaustively on all paths Q. If Reduction Rule 4.9 is applicable on Q, then we replace Q with a simple NO-instance. This takes constant time. Thus, applying Reduction Rule 4.9 on all paths has a running time of $\mathcal{O}(n)$.

(3) Next we apply Reduction Rule 4.10 exhaustively. Thus, we have to delete all labeled inner nodes and their left neighbors in all paths. This takes $\mathcal{O}(n)$ time. Then, for all left neighbors p_{left} of labeled nodes in paths we have to add all their neighbors $v \in N_G(p_{\text{left}})$ to the list F of nodes that have to be matched delete at most all edges incident to v. This has a running time of $\mathcal{O}(\sum_{u \in V(G)} \deg(u)) = \mathcal{O}(m)$. After we have applied this data reduction rule on a path Q, we are again left with a degree-one node. As described in point (1) we then apply the degree-one data reduction rules repetitively on all new degree-one nodes that arise, which can be done in $\mathcal{O}(m)$ for all paths where Reduction Rule 4.10 is applied. This could leave us with two paths that have to be joined. Doing so for all paths, where Reduction Rule 4.10 is applied leads to a running time of $\mathcal{O}(n)$ Thus, the running time of applying Reduction Rule 4.10 exhaustively is $\mathcal{O}(n + m)$.

(4) At last we apply Reduction Rule 4.11 exhaustively on all remaining paths Q. We delete all but four or five nodes of Q and add one edge. Hence, applying the reduction rules on all paths of the graph, deletes less then n nodes and m edges and adds less then n edges. Thus, applying Reduction Rule 4.11 exhaustively takes $\mathcal{O}(n + m)$ time

Altogether this leads to a running time of $\mathcal{O}(n + m)$ for applying Reduction Rules 4.8 to 4.11 exhaustively. □

Chapter 5

Kernelization

In this chapter, we show that the data reduction rules from Chapter 4 yield a linear-time computable, linear-size kernel for Strongly Stable Matching parameterized by feedback edge number. Recall that to formulate these data reduction rules we introduced an annotation of Strongly Stable Matching, which we called Annotated Strongly Stable Matching. Thus, the kernel which we will derive from these data reduction rules is actually a kernel for Annotated Strongly Stable Matching. To achieve our goal, a linear-time computable kernel for Strongly Stable Matching parameterized by feedback edge number, we also have to formulate a polynomial-time many-to-one reduction from Strongly Stable Matching to Annotated Strongly Stable Matching which can be executed in linear time and vice versa. This is done in Section 5.1. In Section 5.2, we will describe why the data reduction rules formulated in Chapter 4 result in a linear-time computable, linear-size kernel parameterized by feedback edge number for Annotated Strongly Stable Matching and how this leads to a kernel for Strongly Stable Matching.

5.1 Reductions between Strongly Stable Matching and Annotated Strongly Stable Matching

Racall that the data reduction rules from Chapter 4 are all applied on instances of Annotated Strongly Stable Matching. Thus, we cannot directly use them for a kernelization of Strongly Stable Matching. First we have to reduce Strongly Stable Matching to Annotated Strongly Stable Matching. As our goal is to find a linear-size kernel that is linear-time computable this reduction has to run in linear time. We can then apply the data reduction rules from Chapter 4 on the instance of Annotated Strongly Stable Matching. Assuming that the data reduction rules yield a linear-time computable kernel for Annotated Strongly Stable Matching, we are then able to form a linear-size kernel for Strongly Stable Matching if we are able to find a linear-time reduction from Annotated Strongly Stable Matching to Strongly Stable Matching.

We will first formulate the reduction from Strongly Stable Matching to Annotated Strongly Stable Matching. This is very easy, as Annotated Strongly Stable Matching is just an extended version of Strongly Stable Matching.

Lemma 5.1. Strongly Stable Matching *can be reduced to* Annotated Strongly Stable Matching *in linear time such that the instance of* Strongly Stable Matching *and the corresponding reduced instance of* Annotated Strongly Stable Matching *have the same feedback edge number.*

Proof. Let $\mathcal{I} = (G = (V, E), P)$ be an instance of Strongly Stable Matching and $\mathcal{I}' = (G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching where $F := \emptyset$ is the set of nodes that have to be matched and $L := \emptyset$ is the set of labeled nodes in Annotated Strongly Stable Matching. Instances $\mathcal{I}$ and $\mathcal{I}'$ have the same nodes and edges. Thus, their feedback edge number is the same. As $F = \emptyset$ and $L = \emptyset$ the instance $\mathcal{I}'$ of Annotated Strongly Stable Matching is equivalent to instance $\mathcal{I}$ of Strongly Stable Matching, when deleting F and L. Thus, a matching M is strongly stable in $\mathcal{I}$ if and only if M is annotated strongly stable in $\mathcal{I}'$. □

The reduction from Annotated Strongly Stable Matching to Strongly Stable Matching is not quite as simple. In instances of Annotated Strongly Stable Matching the sets F and L exist, which do not exist in instances of Strongly Stable Matching. Thus, we have to find a way to remove those sets, without deleting the information they represent. Recall that our original reason to introduce the set L was to be able to delete degree-one nodes with certain preferences. We can now reinsert the degree-one nodes the labels represent, i.e. for each labeled node we can add a degree-one neighbor as its ℓ^{th} preference. It might seem counter-intuitive to reinsert the nodes we specifically wanted to delete. However, remember that we execute this reduction of Annotated Strongly Stable Matching to Strongly Stable Matching on the linear-size kernel of Annotated Strongly Stable Matching. Thus, the number of nodes and number of edges in the instance of Annotated Strongly Stable Matching is already bounded by the parameter feedback edge number and we are also able to bound the number of added degree-one nodes.

The set F contains nodes that have to be matched in the Annotated Strongly Stable Matching. Thus, the reduction has to accomplish that in the reduced instance a strongly stable matching only exists if all nodes in F can be matched in the original instance of Annotated Strongly Stable Matching. Thus, for all $v \in F$ we add two degree-one neighbors of v that v has as last preference. Hence, these two new neighbors of v only become a possible matching partner if v cannot be matched to any other node. But, as both new neighbors have v as first and only preference, one of the new edges between v and the new neighbors must be blocking if v is not matched to a node that it prefers to its new neighbors. Thus, we have accomplished that v has to be matched.

Lemma 5.2. Annotated Strongly Stable Matching *can be reduced to* Strongly Stable Matching *in linear time.*

Proof. Let $\mathcal{I} = (G = (V, E), F, L, P)$ be an instance of Annotated Strongly Stable Matching. In Annotated Strongly Stable Matching there exist sets F and L, which do not exist in Strongly Stable Matching. Thus, we need to find a way to dismiss F and L, while keeping the constraints these sets pose. Recall that with the graph noL(G) (see Definition 4.1) we already defined a graph, where all labels are deleted.

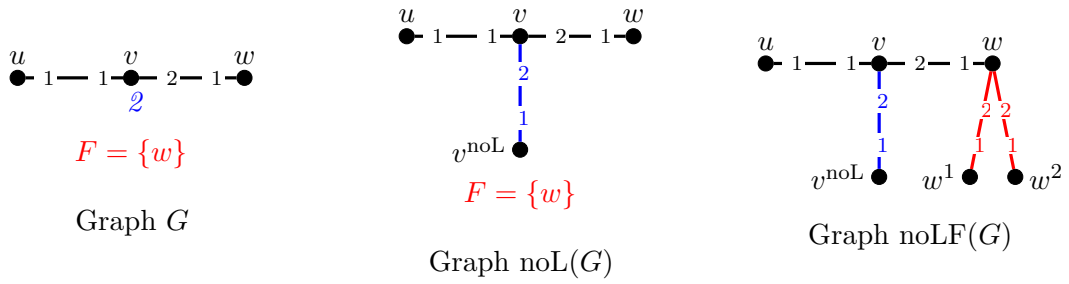


Figure 5.1: An example showing a graph G, its corresponding graph $\text{noL}(G)$ and its corresponding graph $\text{noLF}(G)$. Graph G contains one labeled node v and one node $w \in F$ that has to be matched. First we construct graph $\text{noL}(G)$ by adding one additional neighbor v^{noL} to v. For a more in depth explanation of the construction of $\text{noL}(G)$ look at Figure 4.2. In $\text{noL}(G)$ we still have the set F of nodes that have to be matched. Starting from $\text{noL}(G)$ we construct $\text{noLF}(G)$, where this set does not exist. Node w is element of F in $\text{noL}(G)$. Thus, in $\text{noLF}(G)$ node w has two additional neighbors w^1 and w^2. Those neighbors are w's last preference. As w only has a first preference in $\text{noL}(G)$, the additional neighbors are w's second preference. The new nodes w^1 and w^2 only have one preference, which is node w. Besides w no nodes are in the set F of nodes that have to be matched in $\text{noL}(G)$. Thus, the preference lists of the other nodes remain the same in $\text{noLF}(G)$.

We will now use this already existing definition. We delete the set L of labeled by forming the graph $\text{noL}(G)$. Without loss of generality we say that $V(\text{noL}(G)) = \{v_1, v_2, ..., v_n\}$. We will now define a graph $\text{noLF}(G)$ that further transforms the graph $\text{noL}(G)$, to also delete the set F of nodes that have to be matched. A visualization is provided in Figure 5.1.

$$\text{noLF}(G) := (V(\text{noL}(G)) \cup \{v_i^1, v_i^2 \mid v_i \in F\}, E(\text{noL}(G)) \cup \{\{v_i, v_i^1\}, \{v_i, v_i^2\} \mid v_i \in F\})$$

Furthermore, the preference list $P_{\text{noLF}(G)}(u)$ for each node $u \in V(\text{noLF}(G))$ is:

$$P_{\text{noLF}(G)}(u) := \begin{cases} P_{\text{noL}(G)}(u), & \text{if } u \in V(\text{noL}(G)) \setminus F, \\ P_{\text{noL}(G)} \circ (\{v_i^1, v_i^2\}), & \text{if } u = v_i \in F, \\ v_i, & \text{if } u = v_i^1 \vee u = v_i^2. \end{cases}$$

Now we have to prove that a matching M is annotated strongly stable for the instance $\mathcal{I}$ if and only if there exists a matching M' that is strongly stable for the instance $\mathcal{I}' = (\text{noLF}(G), P_{\text{noLF}(G)})$.

In Definition 4.2 we already defined a matching $M_{\text{noL}} = M \cup \{\{v, v^{\text{noL}}\} \mid v \in L : \forall e \in M : v \notin e\}$ which is strongly stable in $\text{noL}(G)$. We will show that M_{noL} is strongly stable in $\mathcal{I}'$ if and only if M is annotated strongly stable in $\mathcal{I}$.

First we will show that if M is an annotated strongly stable matching in $\mathcal{I}$, then M_{noL} is a strongly stable matching in $\mathcal{I}'$.

We start by looking at the nodes $v \in L$. If a node $v \in L$ is matched in M, then it must be matched to a node $u \in V(G)$ which it prefers to its label. Node u still exists in $\text{noLF}(G)$ and v can also be matched to u in the instance $\mathcal{I}'$ as is satisfied by M_{noL}.

Moreover, no edge is blocking in graph noLF(G) if v is matched to u, as v can only have three neighbors in noLF(G) which it did not already have in G. Node v must have one additional neighbor $v^{\text{noL}} \in V(\text{noLF}(G)) \setminus V(G)$ which v likes exactly as much as it likes its label in $\mathcal{I}$. Hence, v likes v^{noL} less than u. Node v might have two more additional neighbors in $V(\text{noLF}(G)) \setminus V(G)$ due to the fact that v might be in F. However, if v has these two additional neighbors, then they must be the last preference of v. Thus, they cannot cause blocking edges.

If node v is not matched in M, then all of v's neighbors which v prefers to its label must be matched to a node they prefer to v. All of v's neighbors which v likes as much as its label must be matched to a node they like at least as much as v. All these neighbors as well as their neighbors still exist in noLF(G) as no nodes and edges are deleted in the reduction. Thus, these nodes can be matched in the same way as in M. The only difference is that v is matched to its new neighbor v^{noL} which it likes exactly as much as its label, as is the case in M_{noL}.

Now we look at the list F of nodes that have to be matched in M. Let $v_i \in F$ be a node that has to be matched in G. Assume that v_i is not labeled. Then v_i must be element of a matching edge in M, without having blocking edges. Thus, it is also possible to match v in M_{noL}, as all nodes and edges that exist in G also exist in noLF(G). Moreover, no blocking edges are produced by the additional nodes and edges in noLF(G). In noLF(G), v_i has two more neighbors v_i^1 and v_i^2 which v_i likes less than all other neighbors. Thus, as $v_i \in F$ must be matched, the edges $\{v_i, v_i^1\}$ and $\{v_i, v_i^2\}$ cannot be blocking. All edges not incident to a node in $F \cup L$ are not blocking as their endpoints are matched the same in M and M_{noL}. Thus, if an annotated strongly stable matching M exists in $\mathcal{I}$, then M_{noL} is a strongly stable matching in $\mathcal{I}'$.

Now we show that if a strongly stable matching M' exists in $\mathcal{I}'$, then $M = M' \setminus \{\{v, v^{\text{noL}}\} \mid v \in L\}$ is strongly stable in $\mathcal{I}$. Again, we first look at the labeling. In noLF(G) no labels exists. But all nodes v which are labeled in the instance $\mathcal{I}$ have one additional neighbor v^{noL} in noLF(G), which they like exactly as much as their label in $\mathcal{I}$. Thus, if v is matched to v^{noL} in M', then analogously to the other direction, this means that all neighbors of v which v prefers to v^{noL} must be matched to a node they prefer to v and all neighbors of v which v likes exactly as much as v^{noL} must be matched to a node they like at least as much as v. As all these nodes also exist in the instance $\mathcal{I}$, they must be matched the same in the matching M to cause no blocking edges. Thus, node v does not cause blocking edges.

If v is not matched to v^{noL} in the matching M' in instance $\mathcal{I}'$, then v must be matched to a neighbor it prefers to v^{noL} for a strongly stable matching to exist. The only other neighbors v can have in noLF(G) which do not exist in G must be the last preference of v. Thus, node v cannot be matched to them as this would cause $\{v, v^{\text{noL}}\}$ to be blocking. Hence, in this case node v is matched to the same node in M and M'.

Now we look at the set F. Let $v_i \in F$ be a node that has to be matched in an annotated strongly stable matching in $\mathcal{I}$. Then v has two additional neighbors v_i^1 and v_i^2 in noLF(G). Both neighbors are v_i's last preference. Node v_i must be matched to a node that is not one of those additional neighbors, i.e., a node that also exists in G. Otherwise one of the edges $\{v_i, v_i^1\}$ or $\{v_i, v_i^2\}$ would be blocking. If for example v_i would be matched to v_i^1, then $\{v_i, v_i^2\}$ would be blocking, as v_i is indifferent between v_i^1 and v_i^2

and v_i^2 cannot be matched to a node besides v_i as v_i is v_i^2's only neighbor. Thus, node v_i must be matched to a node in M' which also exists in the graph G. In I node v must also be matched as v is in F. In the matching M in $\mathcal{I}$ node v can be matched to the same node as it is matched to in M' in $\mathcal{I}'$

Again, all edges not incident to a node in $F \cup L$ are not blocking as their endpoints are matched the same in M and M'. Hence, altogether if a strongly stable matching M' in $\mathcal{I}'$ exists, then M is annotated strongly stable in $\mathcal{I}$. □

5.2 Linear-size Kernel for Strongly Stable Matching Parameterized by Feedback Edge Number

Mertzios, Nichterlein, and Niedermeier [MNN20] provided a linear-time computable linear-size kernel for Maximum Matching parameterized by the feedback edge number. As we did in Chapter 4, Mertzios, Nichterlein, and Niedermeier also fomulated data reduction rules to delete all degree-zero and degree-one nodes, as well as data reduction rules to delete degree-two nodes. While we could not adopt their data reduction rules due to the different requirements of Strongly Stable Matching in comparison to Maximum Matching, we can use a similar reasoning to argue why the data reduction rules from Chapter 4 lead to a linear-size kernel for Annotated Strongly Stable Matching with parameter feedback edge number. To prove the following theorem, we will adopt their proof to our data reduction rules.

Theorem 5.3. Annotated Strongly Stable Matching *admits a linear-time computable linear-size kernel with respect to the parameter feedback edge number.*

Proof. Apply Reduction Rules 4.2 to 4.11 in linear time, as stated by Lemmas 4.14 and 4.24. Let $G = (V, E)$ be the reduced graph and $X \subseteq E$ be the feedback edge set with $|X| \leq k$. Furthermore, let $V^1_{G'}, V^2_{G'}$ and $V^{\geq 3}_{G'}$ be the nodes that have degree one, two, and three or more in the graph $G' = (V' = V, E' = E \setminus X)$, which is a forest. As Reduction Rules 4.2 to 4.11 are not applicable in G, a node $v \in V$ with $\deg_G(v) = 1$ does not exist. Thus, in graph G all nodes $v \in V^1_{G'}$ have to be incident to an edge from X. As each edge can be incident to at most two nodes and $|X| \leq k$, we can conclude that $|V^1_{G'}| \leq 2k$.

By definition, G' has a feedback edge number of zero. Thus, graph G' is a forest. Hence, we know that $|V^{\geq 3}_{G'}| < |V^1_{G'}|$ and thus $|V^{\geq 3}_{G'}| < 2k$.

Now we have to look at $V^2_{G'}$. A node of degree-two in graph G' must either be also of degree-two in graph G, or incident to at least one edge from the feedback edge set X. As $|X| \leq k$ and each edge can be incident to at most two nodes, there can be a maximum of $2k$ nodes that are of degree two in G' but not in G. If the node is also of degree-two in graph G, then it must be an inner node of a path in G. As Reduction Rules 4.2 to 4.4 and 4.8 to 4.11 are not applicable, all paths in G can contain at most 13 nodes of which 11 nodes are inner nodes, as shown in Observation 4.22. Furthermore, as no degree-one nodes exist in G, the outer nodes of the paths must have at least degree three in G (if they had degree two, then they would actually be inner nodes of the path). In graph G the number of nodes of degree at least three is at most $|V^{\geq 3}_{G'}| + 2k \leq 4k$, as each of the at most k edges in the feedback edge set X can have a maximum of two incident nodes.

As each path has two outer nodes, there can be at most $\frac{4k}{2} = 2k$ paths in G. Reasoning that all degree two nodes are inner nodes of paths or incident to at least one edge from the feedback edge set, we can conclude that $|V^2_{G'}| \leq 2k \cdot 11 + 2k = 24k$.

As we do not delete any nodes, but only edges in the construction of G' the number of nodes in G and G' is equal. Thus, the total number of vertices in G is $|V^1_{G'}|+|V^2_{G'}|+|V^{\geq 3}_{G'}| \leq 28k$. As G' is a forest, we can also conclude that the number of edges in G is at most $|V| + k \leq 29k$. □

Given the linear-size kernel for Annotated Strongly Stable Matching, we can now conlclude with the fomulation of the kernel for Strongly Stable Matching resulting in the main theorem of this thesis:

Theorem 5.4. Strongly Stable Matching *admits a linear-time computable linear-size kernel with respect to the parameter feedback edge number k.*

Proof. Let $\mathcal{I} = (G = (V, E), P)$ be an instance of Strongly Stable Matching. As Lemma 5.1 states, we can reduce instance $\mathcal{I}$ to an instance $\mathcal{I}_{\text{ASSM}} = (G = (V, E), F, L, P)$ of Annotated Strongly Stable Matching in linear time. Moreover, Lemma 5.1 states that the feedback edge numbers in $\mathcal{I}$ and $\mathcal{I}_{\text{ASSM}}$ are the same.

As proven in Theorem 5.3, Annotated Strongly Stable Matching admits a linear-size kernel parameterized by feedback edge number k. More specifically, we proved that graph G can contain at most $28k$ nodes and $29k$ edges.

In Lemma 5.2 we showed that we can reduce Annotated Strongly Stable Matching to Strongly Stable Matching in linear time. Let $\mathcal{I}' = (G' = (V', E'), P')$ be the instance of Strongly Stable Matching which results from reducing $\mathcal{I}_{\text{ASSM}}$. As the reduction from Annotated Strongly Stable Matching to Strongly Stable Matching is linear-time computable, it follows that $\mathcal{I}'$ must be a linear-size kernel.

We will now further specify the exact size of the kernel for Strongly Stable Matching. In the reduction we added one node v^{noL} for every node $v \in L$ to the graph and two nodes v^1 and v^2 for every node $v \in F$ to the graph. L and F can contain no more than all nodes in the graph. Thus, during the reduction at most three nodes get added for every node contained in the graph. Hence, we can conclude that $|V'| \leq 4|V| \leq 4 \cdot 28k = 112k$. For every node that is added during the reduction, we add exactly one edge. Thus, we can also conclude that $|E'| \leq |E| + 3|V| \leq 29k + 3 \cdot 28k = 113k$

Altogether, this leads to a linear-time computable, linear-size kernel for Strongly Stable Matching parameterized by feedback edge number k. □

Chapter 6

Conclusion

In this thesis, we investigated the parameterized complexity of Strongly Stable Matching. We showed that it admits a linear-size, linear-time computable problem kernel parameterized by the feedback edge number. To achieve the linear-time computable problem kernel, we developed data reduction rules to remove degree-zero and degree-one nodes. Furthermore, we developed data reduction rules to reduce all paths of degree-two nodes to a constant length. We showed that altogether the data reduction rules can be applied exhaustively in linear time. In the process of developing the data reduction rules to delete all degree-one nodes we came across several issues. In order to overcome them we defined an annotation of Strongly Stable Matching, which we called Annotated Strongly Stable Matching. While some of the data reduction rules might also directly work on instances of Strongly Stable Matching, we only developed them for instances of Annotated Strongly Stable Matching, as this is sufficient for our goal. The data reduction rules enable us to form a linear-size, linear-time computable kernel for Annoated Strongly Stable Matching. We then provided a linear-time computable polynomial-time many-to-one reduction from Strongly Stable Matching to Annotated Strongly Stable Matching and vice versa. We finished with the main theorem of this thesis, a linear-size, linear-time computable kernel for Strongly Stable Matching.

Feedback edge number is the only parameter we considered. Other parameters could be considered as well. The feedback edge number gives information connected to the number of cycles in a graph as well as its treelikeness. While the parameter *feedback vertex number* does not directly give information about the number of cycle in the graph it does give information about its treelikeness. The feedback vertex number of a graph is the number of vertices/nodes that would have to be deleted from a graph to obtain a tree or forest. Thus, the feedback vertex number is upper-bounded by the feedback edge number of a graph. The parameter *tree-width* is an indicator of the number of cycles in a graph and upper-bounded by the feedback vertex number plus one. Because of this connection between feedback edge number, feedback vertex number and tree-width it would be interesting to investigate whether it is possible to extend our set of data reduction rules, such that we can reuse them to develop linear-time computable problem kernels regarding the feedback vertex number or tree-width.

In real-world applications, we are often presented with many constraints that can be

modeled using weights. Thus, it would also be interesting to investigate parameterizations of the weighted version of Strongly Stable Matching.

We could also further validate the data reduction rules developed in this thesis by testing them in practical implementation.

Chapter 7

Acknowledgements

First of all, I would like to thank Professor Dr. Niedermeier for giving me the opportunity to write this bachelor thesis under his supervision. I would like to thank him for always giving very quick and reliable feedback and advice. Furthermore, I would like to thank my supervisors Klaus Heeger and Dr. André Nichterlein for their constant and excellent support and feedback. They were always available to discuss problems and new ideas, help me finding even the most inconspicuous *edge cases* (hopefully), and they also guided me in writing my thesis. Last but not least, I would like to thank my family for patiently supporting and caring for me while I was preparing and writing this thesis.

Literature

[Adi+18] D. Adil, S. Gupta, S. Roy, S. Saurabh, and M. Zehavi. "Parameterized algorithms for stable matching with ties and incomplete lists". In: *Theoretical Computer Science* 723 (2018), pp. 1–10 (cit. on p. 14).

[Bre+19] R. Bredereck, K. Heeger, D. Knop, and R. Niedermeier. "Parameterized Complexity of Stable Roommates with Ties and Incomplete Lists Through the Lens of Graph Parameters". In: *30th International Symposium on Algorithms and Computation (ISAAC 2019)*. Vol. 149. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 44:1–44:14 (cit. on p. 14).

[CJ21] A. Cseh and A. Juhos. "Pairwise Preferences in the Stable Marriage Problem". In: *ACM Transactions on Economics and Computation* 9.1 (Jan. 2021) (cit. on p. 13).

[CKK19] Y.-K. Che, J. Kim, and F. Kojima. "Stable Matching in Large Economies". In: *Econometrica* 87.1 (2019), pp. 65–110 (cit. on p. 10).

[Cyg+15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015 (cit. on p. 19).

[Edm65] J. Edmonds. "Paths, Trees, and Flowers". In: *Classic Papers in Combinatorics*. Boston, MA: Birkhäuser Boston, 1965, pp. 361–379 (cit. on p. 12).

[Fed92] T. Feder. "A New Fixed Point Approach for Stable Networks and Stable Marriages". In: *Journal of Computer and System Sciences* 45.2 (1992), pp. 233–284 (cit. on p. 13).

[Gab85] H. N. Gabow. "A scaling algorithm for weighted matching on general graphs". In: *26th Annual Symposium on Foundations of Computer Science (FOCS 1985)*. Portland, OR, USA: IEEE Computer Society, 1985, pp. 90–100 (cit. on p. 12).

[Gha+15] M. Gharote, R. Patil, S. Lodha, and R. Raman. "Assignment of trainees to software project requirements: A stable matching based approach". In: *Computers & Industrial Engineering* 87 (2015), pp. 228–237 (cit. on p. 10).

[Grü11] D. N. zu Grünberg. "Die Vergabe von Studienplätzen durch die Stiftung für Hochschulzulassung–Eine Analyse der Anforderungen an öffentlich-rechtliche Stiftungen im Bildungswesen". In: *RdJB Recht der Jugend und des Bildungswesens* 59.3 (2011), pp. 370–384 (cit. on p. 9).

[GS62] D. Gale and L. S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 120.5 (1962), pp. 386–391 (cit. on pp. 10, 12, 13).

[GSZ17] S. Gupta, S. Saurabh, and M. Zehavi. “On Treewidth and Stable Marriage”. In: *Computing Research Repository* abs/1707.05404 (2017). arXiv: 1707.05404 (cit. on p. 13).

[HK73] J. E. Hopcroft and R. M. Karp. “An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs”. In: *SIAM Journal on Computing* 2.4 (Dec. 1, 1973), pp. 225–231 (cit. on p. 12).

[Hua10] C. Huang. “Circular Stable Matching and 3-way Kidney Transplant”. In: *Algorithmica* 58.1 (2010), pp. 137–150 (cit. on p. 10).

[IM02] R. W. Irving and D. F. Manlove. “The Stable Roommates Problem with Ties”. In: *Journal of Algorithms* 43.1 (Apr. 1, 2002), pp. 85–105 (cit. on pp. 12, 13).

[Irv85] R. W. Irving. “An Efficient Algorithm for the ”Stable Roommates” Problem”. In: *Journal of Algorithms* 6.4 (1985), pp. 577–595 (cit. on p. 12).

[Irv94] R. W. Irving. “Stable Marriage and Indifference”. In: *Discret. Appl. Math.* 48.3 (1994), pp. 261–272 (cit. on pp. 12, 13).

[Kav+04] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. “Strongly Stable Matchings in Time $\mathcal{O}(nm)$ and Extension to the Hospitals-Residents Problem”. In: *STACS 2004*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 222–233 (cit. on p. 12).

[KK12] Y. Kamada and F. Kojima. “Stability and Strategy-Proofness for Matching with Constraints: A Problem in the Japanese Medical Match and Its Solution”. In: *American Economic Review* 102.3 (2012), pp. 366–70 (cit. on p. 10).

[Kor+18] V. Korenwein, A. Nichterlein, R. Niedermeier, and P. Zschoche. “Data Reduction for Maximum Matching on Real-World Graphs: Theory and Experiments”. In: *26th Annual European Symposium on Algorithms, (ESA 2018), August 20-22, 2018, Helsinki, Finland*. Vol. 112. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 53:1–53:13 (cit. on p. 14).

[Kun16] A. Kunysz. “The Strongly Stable Roommates Problem”. In: *24th Annual European Symposium on Algorithms (ESA 2016)*. Vol. 57. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 60:1–60:15 (cit. on pp. 12, 13, 21).

[Kun18] A. Kunysz. “An Algorithm for the Maximum Weight Strongly Stable Matching Problem”. In: *29th International Symposium on Algorithms and Computation (ISAAC 2018)*. Vol. 123. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 42:1–42:13 (cit. on p. 13).

[Man13] D. Manlove. *Algorithmics of matching under preferences*. Vol. 2. World Scientific, 2013 (cit. on p. 12).

[MNN20] G. B. Mertzios, A. Nichterlein, and R. Niedermeier. "The Power of Linear-Time Data Reduction for Maximum Matching". In: *Algorithmica* 82.12 (2020), pp. 3521–3565 (cit. on pp. 14, 59).

[MR20] K. Meeks and B. Rastegari. "Solving hard stable matching problems involving groups of similar agents". In: *Theoretical Compututer Science* 844 (2020), pp. 171–194 (cit. on p. 14).

[MS10] D. Marx and I. Schlotter. "Parameterized Complexity and Local Search Approaches for the Stable Marriage Problem with Ties". In: *Algorithmica* 58.1 (2010), pp. 170–187 (cit. on p. 13).

[MV80] S. Micali and V. V. Vazirani. "An $\mathcal{O}(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs". In: *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*. IEEE Computer Society, 1980, pp. 17–27 (cit. on p. 12).

[Ron90] E. Ronn. "NP-complete stable matching problems". In: *Journal of Algorithms* 11.2 (June 1, 1990), pp. 285–304 (cit. on pp. 12–14).

[Ste] D. P. Stegelmann. *Bewerbung um einen Studienplatz Medizin*. URL: https://www.studieren-medizin.de/9,1,bewerbung_deutschland.html. (accessed: 03.03.2021) (cit. on p. 9).

[Zha10] H. Zhang. "Analysis of the Chinese college admission system". PhD thesis. The University of Edinburgh, Dec. 2010 (cit. on pp. 10, 14).