

DEGREE HEURISTICS FOR MATCHING

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Goethe-Universität
in Frankfurt am Main

von
Bert Besser
aus Leipzig

Frankfurt 2016
(D 30)

vom Fachbereich Informatik und Mathematik der
Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Uwe Brinkschulte

Gutachter: Prof. Dr. Georg Schnitger, Prof. Dr. Ulrich Meyer

Datum der Disputation: 5.7.2016

Abstract

Algorithms for the Maximum Cardinality Matching Problem which greedily add edges to the solution enjoy great popularity. We systematically study strengths and limitations of such algorithms, in particular of those which consider node degree information to select the next edge.

Concentrating on nodes of small degree is a promising approach: it was shown, experimentally and analytically, that very good approximate solutions are obtained for restricted classes of random graphs. Results achieved under these idealized conditions, however, remained unsupported by statements which depend on less optimistic assumptions.

The KARPSSIPER algorithm and 1-2-GREEDY, which is a simplified variant of the well-known MINGREEDY algorithm, proceed as follows. In each step, if a node of degree one (resp. at most two) exists, then an edge incident with a minimum degree node is picked, otherwise an arbitrary edge is added to the solution.

We analyze the approximation ratio of both algorithms on graphs of degree at most Δ . Families of graphs are known for which the expected approximation ratio converges to $\frac{1}{2}$ as $\Delta \rightarrow \infty$, even if randomization against the worst case is used. If randomization is not allowed, then we show the following convergence to $\frac{1}{2}$: the 1-2-GREEDY algorithm achieves approximation ratio $\frac{\Delta-1}{2\Delta-3}$; if the graph is bipartite, then the more restricted KARPSSIPER algorithm achieves the even stronger factor $\frac{\Delta}{2\Delta-2}$.

These guarantees set both algorithms apart from other famous matching heuristics like e.g. GREEDY or MRG: these algorithms depend on randomization to break the $\frac{1}{2}$ -barrier even for paths with $\Delta = 2$. Moreover, for any Δ our guarantees are strictly larger than the best known bounds on the expected performance of the randomized variants of GREEDY and MRG.

To investigate whether KARPSSIPER or 1-2-GREEDY can be refined to achieve better performance, or be simplified without loss of approximation quality, we systematically study entire classes of deterministic greedy-like algorithms for matching. Therefore we employ the *adaptive priority algorithm framework* by Borodin, Nielsen, and Rackoff: in each round, an adaptive priority algorithm requests one or more edges by formulating their properties—like e.g. “is incident with a node of minimum degree”—and adds the received edges to the solution. No constraints on time and space usage are imposed, hence an adaptive priority algorithm is restricted only by its nature of picking edges in a greedy-like fashion.

If an adaptive priority algorithm requests edges by processing degree information, then we show that it does not surpass the performance of KARPSSIPER: our $\frac{\Delta}{2\Delta-2}$ -guarantee for bipartite graphs is tight and KARPSSIPER is optimal among all such

“degree-sensitive” algorithms even though it uses degree information merely to detect degree-1 nodes.

Moreover, we show that if degrees of *both* nodes of an edge may be processed, like e.g. the DOUBLE-MINGREEDY algorithm does, then the performance of KARPSSIPER can only be increased marginally, if at all.

Of special interest is the capability of requesting edges not only by specifying the degree of a node but additionally its set of neighbors. This enables an adaptive priority algorithm to “traverse” the input graph. We show that on general degree-bounded graphs no such algorithm can beat factor $\frac{\Delta-1}{2\Delta-3}$. Hence our bound for 1-2-GREEDY is tight and this algorithm performs optimally even though it ignores neighbor information.

Furthermore, we show that an adaptive priority algorithm deteriorates to approximation ratio exactly $\frac{1}{2}$ if it does not request small degree nodes. This tremendous decline of approximation quality happens for graphs on which 1-2-GREEDY and KARPSSIPER perform optimally, namely paths with $\Delta = 2$. Consequently, requesting small degree nodes is vital to beat factor $\frac{1}{2}$.

Summarizing, our results show that 1-2-GREEDY and KARPSSIPER stand out from known and hypothetical algorithms as an intriguing combination of both approximation quality and conceptual simplicity.

Contents

1	Introduction	13
1.1	Greedy Algorithms for Matching	15
1.2	Focus of the Thesis	18
1.2.1	Degree Information	19
1.2.2	Traversing the Graph	23
1.3	Structure of the Thesis and Credits	24
1.4	Related Work	25
1.4.1	Greedy Algorithms: Experimental Studies	25
1.4.2	Greedy Algorithms: Analytical Studies	26
1.4.3	Exact Algorithms	31
2	Performance Guarantees: Basics	33
2.1	Alternating Paths and Cycles	35
2.1.1	Evolution of Components	36
2.2	Amortization	37
2.2.1	Local Approximation Ratios	37
2.2.2	Transfers	39
2.3	Isolated Nodes	43
3	Performance Guarantees for 1-2-GREEDY	45
3.1	Maximum Degree $\Delta = 3$	46
3.1.1	Saving Coins	47
3.1.2	Amortization for Paths	51
3.2	Maximum Degree $\Delta \geq 4$	56
3.2.1	Canceling Transfers	58
3.2.2	Donations	61
3.2.3	Debts of an M -Edge	65
3.2.4	Amortization for Paths	72
4	Performance Guarantees for KARPSSIPER	83
4.1	Overview of Modifications	83
4.2	Paths	87
4.2.1	New Types of Donations	88
4.2.2	Steps for Path Endpoints	93
4.3	All Components are Balanced	96

5	Approximation Bounds for Greedy Matching	105
5.1	The Priority Algorithm Framework	105
5.1.1	Obtaining Inapproximability Results	109
5.2	Adaptive Priority Algorithms for Matching	110
5.2.1	Degree Sensitive Algorithms	112
5.2.2	Degree Sensitivity for Multiple Nodes	121
5.2.3	Degree Ignorance	123
5.2.4	Hypergraph Matching	124
5.3	Inapproximability Results	125
5.3.1	Proof Techniques	127
5.3.2	Proofs	129
6	Conclusions	155
	References	157
	Appendix A. Linear Time Implementations	163
A.1	GREEDY	164
A.2	MINGREEDY and DOUBLE-MINGREEDY	165
A.3	The KARPSSIPER Algorithm and 1-2-GREEDY	167
	Zusammenfassung	169
	Curriculum vitae	177

List of Figures

1	Basics: Notation Conventions	13
2	Basics: Steps for Path Endpoints	36
3	1-2-GREEDY: Guaranteed F -Edges	40
4	1-2-GREEDY: The Minimum Number of Coins Owned by a Path	43
5	1-2-GREEDY: A $\frac{1}{2}$ -Path Paying a Transfer	53
6	1-2-GREEDY: A Step for an Endpoint (I)	53
7	1-2-GREEDY: A Step for an Endpoint (II)	54
8	1-2-GREEDY: A Step for an Endpoint (III)	55
9	1-2-GREEDY: Transfers are Insufficient to Analyze $\Delta \geq 4$	56
10	1-2-GREEDY: Initiating a Donation	62
11	1-2-GREEDY: A Singleton Saves Two Coins	67
12	1-2-GREEDY: Debts of an M -Edge	67
13	1-2-GREEDY: Sets of Path Endpoints	70
14	1-2-GREEDY: No Degree-1 Endpoint Exists after Creation	75
15	1-2-GREEDY: An Endpoint Receives Two Coins	77
16	1-2-GREEDY: The Second Step for an Endpoint	78
17	1-2-GREEDY: A Dynamic Donation	79
18	KARPSIPSER: Insufficient Transfers for $\Delta = 3$	85
19	KARPSIPSER: Naming Conventions	88
20	KARPSIPSER: Receiving and Paying Two Donations	92
21	KARPSIPSER: Revenues of an Endpoint	93
22	KARPSIPSER: A Step for an Endpoint (I)	95
23	KARPSIPSER: A Step for an Endpoint (II)	96
24	KARPSIPSER: Paths are Balanced (Case 1)	99
25	KARPSIPSER: Paths are Balanced (Case 2)	100
26	KARPSIPSER: Paths are Balanced (Case 4)	101
27	KARPSIPSER: A Singleton Pays Only Transfers	103
28	KARPSIPSER: A Singleton Pays a Donation and Transfers	103
29	Adaptive Priority Algorithms: Inclusion Relations	111
30	\mathcal{DS}_{01} -Algorithms: The Chain of Traps	131
31	\mathcal{DS}_{01} -Algorithms: A Trap for $\Delta = 3$	134
32	\mathcal{VA} -Algorithms: An Extra Component	136
33	\mathcal{VA} -Algorithms: The Center and Gadgets of a Hard Instance	137
34	\mathcal{VA} -Algorithms: A Bipartite Extra Component	143

35	\mathcal{WA} -Algorithms: A Bipartite Gadget	143
36	MDS: A Hard Instance	145
37	\mathcal{DSE}_{01} -Algorithms: The Chain of Traps for $\Delta = 3$	146
38	\mathcal{DSE}_{01} -Algorithms: An Extra Component for $\Delta = 3$	146
39	\mathcal{DSE}_{01} -Algorithms: A Hard Instance for $\Delta \geq 4$	148
40	\mathcal{DSP}_{01} -Algorithms: The Center of a Hard Instance	149
41	\mathcal{DSP}_{01} -Algorithms: Adding a Path	149
42	\mathcal{WA}^k -Algorithms: A Hard Hypergraph Instance	152
43	\mathcal{WA}^k -Algorithms: Definition of Sets S_i	152
44	Linear Time Implementations: Linked Adjacency Arrays	164
45	Linear Time Implementations: Tracking Degrees	166

1 Introduction

So called *matching problems* occur in a variety of applications such as computing block triangular forms of sparse matrices [PF90], image feature matching [CWC⁺96], partitioning graphs to set up compute clusters for large scale computation tasks [KK98], kidney exchange [RSÜ05, Tri12], network traffic routing [HS07], protein structure comparison [BSX08], or empirical studies [LGXB11].

At the core of such applications lies the task to find an allocation of e.g. processors to compute clusters, kidney donors to patients, incoming network packets to output ports of networking switches, or matrix rows to matrix columns. Such tasks can be modeled as finding large sets of node-disjoint edges, where the application at hand determines properties of the underlying graph (e.g. whether it is bipartite or weighted).

In this thesis we study the fundamental Maximum Cardinality Matching Problem, to which we refer as *matching* throughout. Given an undirected unweighted graph $G=(V, E)$, a *maximum matching* is to be determined, cf. Figure 1:

- if $M \subseteq E$ is a set of edges and node $v \in V$ is incident with an edge in M , then we say that v is *matched in M* ;
- if each node in V is matched in at most one edge in M , then M is called a *matching*;
- if no edge can be added to M without matching a node more than once, then M is called a *maximal* (or *non-extensible*) matching;
- if M is at least as large as any other maximal matching in the graph, i.e. if M is a matching of largest possible size, then M is called a *maximum* (or *optimal*) matching.

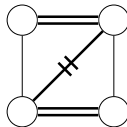


Figure 1: A maximal matching (crossed edge) and a maximum matching (double drawn edges)

Maximum matchings can be computed in polynomial time $O(\sqrt{|V|} \cdot |E|)$, see Section 1.4.3 for an overview of algorithms. However, the employed techniques are sophisticated and non-trivial to implement properly. When approximate matchings serve the purpose good enough, some exact algorithms allow to stop computation prematurely to obtain a (possibly sub-optimal) maximal matching. However, this approach still depends on complex implementations of polynomial time techniques.

Greedy Matching Algorithms. This thesis systematically studies greedy algorithms for matching. Compared to the known polynomial time methods, greedy algorithms are conceptually very simple. Motivation for a greedy approach also arises from the need to compute matchings in scenarios where access to the input is restricted.

Typically, a greedy algorithm starts with an empty matching and repeatedly *picks* an edge which contains only nodes that were not matched in a previously picked edge (refer to Section 1.1 for a more formal description). Characteristically, once an edge is picked, it is not removed from the matching later. The algorithm stops as soon as no further edge can be picked without matching a node twice, i.e. a maximal matching is obtained. Since the algorithm might pick unfavorable edges, the solution is not necessarily optimal.

Experiments suggest that greedy matching algorithms typically produce very large matchings on random graphs and benchmark instances [KS81, Tin84, FRS95, MMH95, Mag98, HS07, LMS10]¹. Furthermore, due to their simplicity, greedy matching algorithms are not hard to implement and mostly run in linear time [KS81, FRS95, Mag98, PS12, BP15]. These qualities render greedy matching algorithms an attractive alternative to exact polynomial time algorithms whenever speed and simplicity are of higher importance than obtaining exact solutions.

Some scenarios impose restrictions on how the input can be accessed, and therefore enforce the greedy construction of a matching. Goel and Tripathi [GT12] discuss the pairwise kidney exchange problem, where pairs of incompatible kidney patients and donors—e.g. family members or married couples—have to be matched such that both patients are able to receive a kidney from the respective other donor. Here, testing two pairs for compatibility is a time-consuming and costly procedure. Therefore once a match is found, the transplant has to be performed. In terms of matching, this problem is modeled as a graph whose nodes represent incompatible patient/donor pairs and edges model mutual compatibilities between such pairs. The edge set is unknown at first, and an algorithm has to “probe” for edges. Once an edge is found, it has to be added to the matching irrevocably.

Besides being interesting in their own right, greedy matching algorithms are often used for preprocessing in exact algorithms. Here, a greedy solution is used as input to an algorithm which iteratively grows the matching. Thus, large greedy matchings can speed up computation considerably [MMH95, LMS10].

¹See Section 1.4.1 for an overview of experimental studies.

Rigorous Analyses of Greedy Matching Algorithms. Over three decades ago, greedy matching algorithms gained popularity when researchers studied algorithmic (instead of non-constructive) approaches to prove the existence of large matchings in random graphs: the idea is to define an algorithm and to show that it almost surely computes a large matching. This line of research was continued for various algorithms, showing that random graphs are solved well [KS81, Tin84, GH90, DFP93, FRS95, AFP98]².

Another fundamental question is how greedy matching algorithms perform for arbitrary graphs. Since any maximal matching is at least half as large as a maximum matching [KH78], the approximation ratio of any algorithm is in the interval $[\frac{1}{2}, 1]$: the factor $\frac{1}{2}$ is a trivial approximation guarantee.

Some algorithms do not beat factor $\frac{1}{2}$, i.e. input graphs can be constructed such that the solution is of size arbitrarily close to half of maximum [DF91, Pol12, BP15]. For those algorithms which do beat factor $\frac{1}{2}$, tight performance bounds remain unknown to date and even the best known bounds leave large gaps to be closed [ADFS95, CCWZ14]. Consequently, researchers also investigated restricted classes of graphs, e.g. trees, planar graphs, graph of high girth, or degree bounded graphs. For these type of graphs stronger bounds could be obtained [DF91, ADFS95, MP97], as we discuss in Section 1.2.

Besides the goal to fully understand particular greedy matching algorithms, there is growing interest into which properties make an algorithm a successful one. Therefore, inapproximability bounds for entire classes of restricted algorithms were established, revealing limits to their approximation performance [GT12, Pol12, BP15]. In particular, none of the most popular representatives computes maximum matchings on general graphs [Pol12, BP15].

1.1 Greedy Algorithms for Matching

Greedy matching algorithms typically follow the scheme given in Algorithm 1, cf. [Tin84]. Starting with an empty matching M , an algorithm proceeds in *steps*, where each step corresponds to one iteration of the loop in Line 2. When the algorithm adds an edge $\{u, v\}$ to M , then we say that the algorithm *picks* edge $\{u, v\}$. From the moment on that edge $\{u, v\}$ is picked, nodes u and v are called *matched*. After u and v are matched, all edges incident with u and v are no longer eligible to be contained in M : therefore these edges are *removed* from the graph, i.e. they are not contained in the graph in future steps. As soon as all edges are removed from the graph, the algorithm outputs M as its solution.

²See Section 1.4.2 for an overview of analytical results.

Algorithm 1 Scheme of a Greedy Matching Algorithm.

1. $M \leftarrow \emptyset$ ▷ initialize empty matching
 2. **while** $E \neq \emptyset$ **do** ▷ as long as there are edges, proceed with next *step*
 3. *select* an edge $\{u, v\} \in E$
 4. $M \leftarrow M \cup \{\{u, v\}\}$ ▷ *pick* the selected edge
 5. $E \leftarrow E \setminus \{\{x, y\} : x \in \{u, v\}, y \in V\}$ ▷ *remove* edges incident with u and v
 6. **end while**
 7. **return** M
-

What distinguishes one algorithm from another is how the next edge is *selected*, i.e. which heuristic is applied in Line 3. In List 1 we present the most popular greedy matching algorithms, where names are taken from the literature.

We prepare the presentation with some basic notation.

The Reduced Graph. Consider the t -th step ($t \geq 1$) of an algorithm, i.e. the t -th iteration of the loop. At the beginning of step t , i.e. when Line 3 is executed for the t -th time, the *reduced graph* G_t denotes the graph from which all edges incident with previously matched nodes are removed. (The input graph is G_1 .) By the *current degree* of node v in step t we refer to the degree of v in the reduced graph G_t , and we denote it as $d_t(v)$. Node v is called *isolated* beginning with the step t' when $d_{t'}(v) = 0$ holds for the first time. Before step t' we call v *non-isolated*.

List 1 (Edge Selection Heuristics).

- i. The GREEDY algorithm, first analyzed by Tinhofer [Tin84], has no preference among edges in the reduced graph and simply selects any one of these.
- ii. The KARPSIPSER algorithm, named after its authors [KS81], proceeds like GREEDY unless a node with current degree one exists: in this case KARPSIPSER selects an edge incident with an arbitrary node of current degree one.

For conciseness, we also call a node of current degree one a *degree-1 node*. For every degree-1 node u there is a maximum matching M^* in the reduced graph such that the edge incident with u belongs to M^* , cf. Proposition 1 in Section 2. Hence picking the edge of a degree-1 node is “optimal”.

- iii. The MRG algorithm [Tin84] (“modified random greedy”, sometimes also called “Simple Greedy”) selects the next edge by first selecting a node u and thereafter selecting one of its neighbors v .

- iv. The MINGREEDY algorithm [Tin84] (sometimes also called “Dynamic MinDegree”) is a variation of MRG which selects the first node u such that it has minimum (but non-zero) current degree among all nodes.

This algorithm in a sense generalizes the approach of the KARPSIPSER algorithm: since for any non-isolated node u there is a maximum matching M^* in the reduced graph such that u is matched in M^* , nodes are preferred for which the chance to select an “optimal” incident edge is as large as possible.

Sometimes MINGREEDY is implemented such that all nodes are sorted in ascending order of degree *once before* the loop in Line 2. In the loop, nodes are then considered in this order, see e.g. [MMH95]. This algorithm is also called “Static MinDegree”. We denote it by STATIC-MINGREEDY.

- v. The SHUFFLE algorithm [GT12] computes a permutation π on all nodes in V before the loop in Line 2. In Line 3 the algorithm selects the π -lexicographically first edge: SHUFFLE selects the first non-isolated node u (according to π) and thereafter selects the first non-isolated neighbor v of u (also according to π).

Compared with MRG, this algorithm does not select an arbitrary neighbor of u but gives higher preference to those neighbors which in previous steps have been tried to be matched without success.

In the literature, algorithms in List 1 are defined such that all ties are broken uniformly at random, i.e. whenever more than one edge, node, or permutation satisfies the respective “definition”, then each is chosen with equal probability. However, in this thesis we also consider the deterministic versions of these algorithms, in which all ties are broken towards the worst case.

The above (randomized) algorithms can be implemented in linear time $O(|V| + |E|)$, see Appendix A. and [KS81, FRS95, Mag98, PS12]. In terms of runtime, this makes them superior to all known exact algorithms³.

List 1 is not complete, since several variations of these algorithms were proposed. In particular, two popular variants of MINGREEDY consider the current degrees of both nodes of an edge:

³See Section 1.4.3 for an overview of efficient algorithms.

List 2 (Variants of MINGREEDY).

- vi. A folklore variant of MINGREEDY is the MDS algorithm (“minimum degree sum”), which selects the next edge $\{u, v\}$ such that the sum of the current degrees of nodes u and v is as small as possible among edges in the reduced graph.
- vii. The “double-sided” (or “two-sided”) variant of MINGREEDY [LMS10] (also formulated in Heuristics 1 and 2 in [Mag98], or called EDSM for “enhanced Degree Sequenced Matching” in [HS07]) first selects a node u of minimum current degree and then a neighbor v of u of minimum current degree. We denote the double-sided MINGREEDY variant as DOUBLE-MINGREEDY.

The “definitions” of algorithms in Lists 1 and 2 are for general graphs. They are typically applied as is to bipartite input graphs, i.e. no attention is paid to the partition from which a node is selected, see e.g. [LMS10].⁴

1.2 Focus of the Thesis

We systematically study deterministic greedy-like algorithms for matching. Our main focus is on strengths and limitations of degree heuristics like e.g. the KARPSIPSER algorithm or MINGREEDY. Therefore we analyze their approximation ratio and compare with the performance of well-known algorithms from the literature as well as with hypothetical algorithms—both more complex ones and conceptually simpler ones.

Our key insight is that KARPSIPSER and (a relaxed version of) MINGREEDY are the conceptually most simple algorithms achieving a non-trivial approximation ratio, and that they perform optimally even among much more sophisticated algorithms.

Gathering Information. How to measure the “sophistication” of an algorithm? The more is known about the input graph, the more fine grained control over the next edge is possible.

Any greedy matching algorithm gathers information about the input graph $G = (V, E)$. E.g. if an algorithm attempts to match nodes u and v but detects that $\{u, v\} \notin E$ holds, then the algorithm infers that u and v have degree at most $|V| - 2$; the MINGREEDY algorithm determines node degrees before the first step and after each step updates

⁴We mention that there are bipartite online scenarios such as AdWord marketing [MSVV07] which require an algorithm to always select a node in the same partition before matching it with a neighbor. The RANKING algorithm of Karp, Vazirani, and Vazirani [KVV90] is one such algorithm, which also inspired the definition of SHUFFLE. However, since the focus of this thesis is on greedy matching, we use the “definitions” given in Lists 1 and 2 throughout.

current degrees of neighbors of newly matched nodes; when the SHUFFLE algorithm probes for neighbors of the “active” node, call it u , then SHUFFLE implicitly learns about edges not incident with u .

Typically, an algorithm does not utilize gathered information entirely. E.g. the GREEDY algorithm simply ignores all gathered information and picks the next edge uniformly at random.

Moreover, in some scenarios only limited information can be obtained. E.g. an algorithm for the pairwise kidney exchange problem (refer to Section 1 for a quick introduction to this problem) has no access to node degrees: for any node the precise number of compatible patient/donor pairs is never determined.

Utilizing Gathered Information. The amount of information gathered in each step limits the “sophistication” of the implemented edge selection heuristic, which in turn determines the approximation quality of a solution. We systematically study what type of information allows for greedy matching heuristics with strong approximation performance, and how well an algorithm can approximate if allowing only restricted information per step. Primarily, we concentrate on two subjects.

What approximation guarantees can be achieved if allowing information on current node degrees? Which “degree heuristic” performs best?

Does information on neighbors of matched nodes—which enables an algorithm to “traverse” the graph—allow for better performance?

In Section 1.2.1 we review known results about “degree-sensitive” algorithms like e.g. MINGREEDY, point out what knowledge gaps remain to be closed, and present our main results. In Section 1.2.2 we discuss algorithms using neighbor information and our related results.

1.2.1 Degree Information

Known results on greedy matching algorithm are mainly for randomized implementations. Even though we study deterministic algorithms in this thesis, in the following overview we refer to randomized versions of the given algorithms. In particular, we highlight strengths and limitations of degree heuristics.

A Benefit on Random Graphs. The GREEDY algorithm, MRG, KARPSIPSER, and MINGREEDY were proven to compute asymptotically optimal matchings on large random graphs [KS81, Tin84, GH90, DFP93, FRS95, AFP98]⁵.

However, experiments suggest that MINGREEDY commonly produces larger matchings than GREEDY and MRG [Tin84, MMH95, Mag98], even by orders of magnitude on random 3-regular graphs [FRS95]⁶. Other experiments indicate that KARPSIPSER and MINGREEDY are capable to find the unique perfect matching⁷ in bipartite graphs whereas GREEDY is not [LMS10].

Performance on Arbitrary Graphs. The GREEDY algorithm implicitly prefers nodes of high degree, since a randomly selected edge is likely to be incident with such a node. This property was exploited by Dyer and Frieze [DF91] in a construction of general graphs: they defined a family of graphs for which GREEDY computes matchings of size arbitrarily close to $\frac{1}{2}$ of optimum.

Giving less importance to high degree nodes allows to break the $\frac{1}{2}$ -barrier. How? The MRG algorithm selects each node with equal probability (and then an incident edge), hence MRG does not prefer large degrees. Aronson, Dyer, Frieze, and Suen [ADFS95] showed that MRG achieves expected approximation ratio at least $\frac{1}{2} + \frac{1}{400.000}$. (In particular, they were the first to obtain a non-trivial bound on the expected approximation ratio of a greedy matching algorithm.) Also, the SHUFFLE algorithm does not prefer large degrees when it initially computes a total order on all nodes. SHUFFLE achieves expected approximation ratio at least $\frac{2(5-\sqrt{7})}{9} \approx 0.523$, as was shown by Chan, Chan, Wu, and Zhao [CCWZ14].

Going one step further and explicitly preferring small degrees, however, lets the approximation performance degrade. Poloczek [Pol12, BP15] showed that MINGREEDY does not beat factor $\frac{1}{2}$. The same construction also shows that DOUBLE-MINGREEDY and MDS do not break the $\frac{1}{2}$ -barrier.

Which type of degree information allows to beat factor $\frac{1}{2}$?

(How) does one of MINGREEDY and KARPSIPSER improve on the performance of the other?

⁵See Section 1.4.2 for an overview of analytical results.

⁶See Section 1.4.1 for an overview of experimental studies.

⁷A matching is called *perfect* if every node is matched.

Restricted Graphs. Hard instances, in particular constructions bounding the approximation ratio to $\frac{1}{2}$, typically contain dense subgraphs with nodes of large degree, see e.g. [DF91, GT12, Pol12, BP15]. Stronger performances could be proven under the assumption that degrees are bounded (as well as for planar graphs, trees, and graphs with large girth) [DF91, ADFS95, MP97]⁸. In particular, both GREEDY and MRG achieve expected approximation ratio at least $\frac{\Delta}{2\Delta-1}$ if the maximum degree in the graph is bounded by at most Δ , as was shown by Dyer and Frieze in [DF91] resp. by Aronson, Dyer, Frieze, and Suen in [ADFS95]. The bound for GREEDY was improved by Miller and Pritikin [MP97] to at least $\frac{1}{2} + \frac{\sqrt{(\Delta-1)^2+1}-(\Delta-1)}{2}$.

What is the approximation performance of KARPSSIPER, MINGREEDY, or DOUBLE-MINGREEDY on degree bounded graphs?

Furthermore, we address these questions:

Can degree information be used in more sophisticated ways than e.g. MINGREEDY to obtain stronger performance?

Is there an algorithm which does not utilize degree information and performs as good as e.g. KARPSSIPER?

Main Results. For graphs of degree at most Δ we analyze the approximation performance of KARPSSIPER, MINGREEDY, and DOUBLE-MINGREEDY, thereby adding to the previously known bounds for GREEDY and MRG.

Worst Case Guarantees. We analyze deterministic variants of these algorithms, i.e. ties are not broken uniformly at random but in worst case fashion. Our guarantees are strictly larger than the best known bounds on the *expected* approximation ratio of GREEDY and MRG (which are $\frac{1}{2} + \frac{\sqrt{(\Delta-1)^2+1}-(\Delta-1)}{2}$ resp. $\frac{\Delta}{2\Delta-1}$, see previous paragraph):

- Bipartite graphs

We show that the KARPSSIPER algorithm achieves approximation ratio at least

$$\frac{\Delta}{2\Delta - 2}.$$

(The same guarantee is implied for MINGREEDY, DOUBLE-MINGREEDY, and any algorithm which in each step selects a node of minimum degree.)

⁸See Section 1.4.2 for an overview of analytical results.

- General graphs

Here, the performance of KARPSSIPER deteriorates to exactly $\frac{1}{2}$, since it might pick the diagonal edge in the graph \mathcal{Z} , which contains a maximum matching of two edges.

We refine the KARPSSIPER algorithm by replacing its edge selection heuristic with the following: if the reduced graph contains a node of degree at most two, select a node of minimum degree and then one of its neighbors; otherwise select an arbitrary edge.

This algorithm, we call it 1-2-GREEDY, achieves approximation ratio at least

$$\frac{\Delta - 1}{2\Delta - 3}.$$

(Observe that 1-2-GREEDY generalizes the MINGREEDY algorithm by relaxing the requirement to select a minimum degree node in case all degrees are at least three.

Again, our guarantees also apply to MINGREEDY, DOUBLE-MINGREEDY, and any algorithm which in each step selects a node of minimum degree.)

Tightness of Guarantees. We contrast both guarantees with tight inapproximability bounds, i.e. the given guarantees cannot be improved. Our bounds also apply to (deterministic versions of) algorithms in Lists 1 and 2 in Section 1.1 as well as further hypothetical algorithms, showing that 1-2-GREEDY and KARPSSIPER perform optimally.

In particular, we analyze classes of deterministic algorithms which are allowed to utilize degree information in *arbitrarily complex* fashion. Therefore we employ the framework of adaptive priority algorithms introduced by Borodin, Nielsen, and Rackoff.

In each round, an algorithm submits a priority order on edges by formulating their characteristics (like e.g. “is incident with a node of minimum degree”), then receives the highest priority edge contained in the graph, and irrevocably adds the received edge to the solution.

There are no resource constraints on the required computations, i.e. the only limitation of an algorithm is its greedy-like nature. Hence inapproximability results apply to correspondingly large classes of algorithms, see Section 5.1 for a detailed discussion.

Further Results. We also investigate a class of more general deterministic adaptive priority algorithms which are allowed to pick entire alternating paths instead of only a single edge. Picking paths, however, does not fundamentally improve approximation performance for general graphs: we show that any such algorithm is bounded by factor $\frac{1}{2}$ for $\Delta \rightarrow \infty$.

1.2.2 Traversing the Graph

The GREEDY algorithm does not utilize degree information to select an edge and might pick—in the worst case—the middle edge of a length-three path, thereby producing a matching with only one edge instead of two, i.e. its approximation ratio is only $\frac{1}{2}$.

Now consider an algorithm A which may (and is restricted to) use degree information in its edge selection routine (as are most of the algorithms presented in Section 1.1). Assume that A selects edge $\{u, v\}$. Since A does not have control over distance or neighbor relationships between u or v and the nodes matched earlier, algorithm A is prone to pick an unfavorable edge: each of u and v might be incident with an edge of a maximum matching. Many unfavorable choices let the approximation performance converge to $\frac{1}{2}$.

This danger is addressed by the following straightforward approach. “Grow” an alternating path by repeatedly picking an edge at one of its ends; if the path cannot be grown any longer, repeat with the next path. This algorithm “traverses” the graph and for this purpose depends on information about neighbors of matched nodes.⁹

If an algorithm is allowed to process neighbor information in arbitrary fashion (e.g. try to construct long alternating paths), can it achieve stronger approximation performance than algorithms restricted to degree information?

Main Results. Our $\frac{\Delta-1}{2\Delta-3}$ -inapproximability bound for general graphs applies not only to “degree-sensitive” algorithms like e.g. 1-2-GREEDY, but to an even larger class of adaptive priority algorithms, namely with additional access to neighbors of selected nodes. In particular, 1-2-GREEDY is optimal within this class, demonstrating the power of using degree information.

Moreover, we show that an algorithm with neighbor information deteriorates to approximation ratio exactly $\frac{1}{2}$ if it does not request nodes of small degree. In particular, this holds for graphs on which (deterministic versions of) KARPSIPSER and 1-2-GREEDY compute maximum matchings, namely paths.¹⁰ Consequently, processing degree information is indispensable in order to surpass factor $\frac{1}{2}$.

Further Results. We also study a class of deterministic adaptive priority algorithms with neighbor information for the more general k -uniform hypergraph matching problem

⁹Angluin and Valiant [AV79] proposed a similar algorithm. However, since their algorithm also removes edges from the matching after they have been picked, we do not consider it a classical greedy algorithm.

¹⁰See Proposition 2 for a formal argument why KARPSIPSER and MINGREEDY perform optimally on paths.

(conventional matching is contained as the special case $k = 2$). This problem is \mathcal{NP} -complete for $k \geq 3$, therefore acceptable approximation performance is not to be expected. We confirm this fact by showing that factor exactly $\frac{1}{k}$ cannot be beaten, even on most simple instances. This performance is worst possible, since any maximal matching has size at least $\frac{1}{k}$ times optimal.

1.3 Structure of the Thesis and Credits

Performance Guarantees. In Section 2 we present the fundamentals of our technique to obtain approximation guarantees for 1-2-GREEDY and the KARPSSIPER algorithm.

In Section 3 we develop guarantees of at least $\frac{\Delta-1}{2\Delta-3}$ for 1-2-GREEDY, where we analyze graphs of maximum degree $\Delta = 3$ and $\Delta \geq 4$ in Section 3.1 resp. Section 3.2. These results improve our previous $\frac{\Delta-1/2}{2\Delta-2}$ -bounds for the MINGREEDY algorithm—first publicly available in [Bes14] and published in [BP15]—in the following two ways. First, 1-2-GREEDY is a “weaker” algorithm, in the sense that in the worst case it performs no better than MINGREEDY. Secondly, the new guarantees for 1-2-GREEDY are larger than the previous ones for MINGREEDY.

In Section 4 we present our $\frac{\Delta}{2\Delta-2}$ -guarantees for the KARPSSIPER algorithm on degree bounded bipartite graphs. These results were first presented in [BW15].

Inapproximability Results. In Section 5.1 we discuss the adaptive priority algorithm framework. In Section 5.2 we first give a rough overview of classes of algorithms analyzed in this thesis and then define them.

Results are presented in Section 5.3. In particular, in Theorem 62 and Theorem 64 we show that algorithms in classes containing KARPSSIPER resp. 1-2-GREEDY do not achieve a stronger approximation performance.

- Theorem 62 improves a result by Bastian Werth presented in [BW15]: for any given maximum degree Δ our inapproximability bound is the same, namely $\frac{\Delta}{2\Delta-2}$, but it applies to a larger class of algorithms. Similarly, Theorem 67 improves another result by Werth.
- Theorem 64 improves joint work with Matthias Poloczek, which was presented first in [Pol12]: for any given maximum degree Δ we improve the inapproximability bound to $\frac{\Delta-1}{2\Delta-3}$, thereby showing that our guarantees for 1-2-GREEDY are tight.

Further results show that processing degree information is indispensable in order to achieve non-trivial approximation performance and that picking paths does not allow to beat factor $\frac{1}{2}$ in general.

Adaptive priority algorithms for k -uniform hypergraph matching fail with approximation ratio exactly $\frac{1}{k}$, as we show in Theorem 70 at the end of Section 5.3.

1.4 Related Work

We present known experimental and analytical results on greedy matching algorithms in Section 1.4.1 resp. Section 1.4.2. For completeness, we also give a brief overview of exact polynomial time algorithms in Section 1.4.3.

1.4.1 Greedy Algorithms: Experimental Studies

Experiments from the literature show that good matching approximations can be obtained with the use of greedy algorithms. However, approximation performance varies noticeably between algorithms. Degree heuristics clearly stand out by commonly producing very large matchings, as we summarize in this section.

Tinhofer [Tin84] compared GREEDY, MRG, and MINGREEDY on random graphs in the Erdős-Rényi model, in which for a given set of nodes the edge between any pair of nodes exists with equal probability p . Independently of the density parameter $0.05 \leq p \leq 0.75$ and the number of nodes (up to 48), GREEDY and MRG performed roughly the same. MINGREEDY performed noticeably better and produced almost optimal matchings.

The same algorithms were compared on random cubic graphs by Frieze, Radcliffe, and Suen [FRS95]. On average, GREEDY and MRG left about the same number of nodes unmatched and MINGREEDY performed best, outperforming the other algorithms by orders of magnitude. In particular, in instances with 10^k nodes both GREEDY and MRG left roughly 10^{k-1} nodes unmatched for $k \in \{2, 3, 4, 5, 6\}$, whereas MINGREEDY left only 10 out of 10^6 nodes unmatched.

MRG and MINGREEDY were also compared by Magun [Mag98] as part of experiments on Erdős-Rényi random graphs with up to 10.000 nodes. Graphs were generated to have small constant average degree between 2 and 6 using edge probability $p = c/|V|$ for various $c \in [2, 6]$. Magun found that for $c \approx 3$ the MRG algorithm produced a number of “lost edges” proportional to the number of nodes (by “lost edges” Magun refers to the difference of the sizes of the solution and a maximum matching). On the other hand, the results indicate that MINGREEDY loses a sub-linear number of edges. Figures in

[Mag98] also suggest that MRG and MINGREEDY perform the better the higher the average degree is.

Möhring and Müller-Hannemann [MMH95] investigated algorithms MRG, MINGREEDY, STATIC-MINGREEDY, and KARPSIPSER when applied to sparse random graphs with up to 2^{15} nodes and average degree between 1 and 10. They found that MINGREEDY and KARPSIPSER computed near optimal matchings of almost equal size. STATIC-MINGREEDY did not perform as well by far, but still noticeably better than MRG.

Langguth, Manne, and Sanders [LMS10] conducted experiments on (sparse) bipartite graphs with a few hundred thousand nodes, where they also investigated the approximation performance of MRG, MINGREEDY, STATIC-MINGREEDY, and the KARPSIPSER algorithm. They investigated two sets of test instances:

- The first test set contained random bipartite graphs, where the model asserts that edges are more evenly distributed than in the Erdős-Rényi model. Here, both MINGREEDY and KARPSIPSER clearly outperformed the other two algorithms and computed very large matching, where on most instances MINGREEDY left slightly less nodes unmatched than the KARPSIPSER algorithm. MRG left roughly three times as many nodes unmatched as STATIC-MINGREEDY.
- On bipartite graphs with a unique perfect matching (a matching is said to be *perfect* if every node is matched), MINGREEDY and KARPSIPSER managed to match every last node. MRG and STATIC-MINGREEDY left about a tenth of the nodes unmatched on average.

Summary. Experimental studies suggest that MINGREEDY and the KARPSIPSER algorithm generally produce very large, even near optimal, matchings. In particular, both performed substantially better than other algorithms. Furthermore, the results indicate that on sparse random graphs MRG and GREEDY perform comparably. In another experiment on sparse random graphs, the STATIC-MINGREEDY algorithm produced better solutions than MRG.

1.4.2 Greedy Algorithms: Analytical Studies

Factor $\frac{1}{2}$ is a trivial guarantee for the approximation ratio of any reasonable matching algorithm. It was studied in the literature how far from $\frac{1}{2}$ and how close to 1 the approximation ratio of particular algorithms is. Moreover, algorithms were compared relative to one another instead of investigating their absolute performance.

In [Tin84], Tinhofer substantiated experimental observations for GREEDY, MRG, and MINGREEDY (on random graphs with constant edge probability p , cf. Section 1.4.1) with analytical results on their performance. He showed that for each given number of nodes $|V|$ there exist intervals such that if $1 - p$ is drawn from these intervals then MINGREEDY performs strictly better than MRG.¹¹

GREEDY. Dyer and Frieze [DF91] established the following monotonicity property for the GREEDY algorithm: if an arbitrary node is removed from the input graph, then the expected size of the matching computed by GREEDY does not increase, and it is decreased by at most one. Using this property, they conducted an analysis depending on the density of a given class of graphs. They showed that GREEDY achieves expected approximation ratio at least $\frac{\Delta}{2\Delta-1}$ on graphs with degree at most Δ . Furthermore, GREEDY's expected approximation ratio is at least $\frac{6}{11}$ (and at most $\frac{11}{15}$) on planar graphs. The expected approximation ratio of GREEDY on forests was determined exactly as 0.7690307...

The most fundamental result in [DF91] is that the expected approximation ratio of GREEDY on general graphs is at most $\frac{1}{2}$, i.e. there is a family of graphs for which, asymptotically, the GREEDY algorithm does not beat the trivial guarantee that holds for any reasonable matching algorithm.

Miller and Pritikin [MP97] bounded the expected approximation ratio of the GREEDY algorithm on planar graphs to at least $\frac{\sqrt{26}-4}{2}$ and at most 0.68436349, thereby improving the bounds in [DF91] at both ends. They also improved Dyer and Frieze's bound for graphs with degree at most Δ to at least $\frac{1}{2} + \frac{\sqrt{(\Delta-1)^2+1}-(\Delta-1)}{2}$. Furthermore, they proved bounds for graphs with girth¹² at least 5: first, GREEDY achieves factor at least $\frac{1}{2} + \frac{1}{2\Delta}$ if any degree is at most Δ ; secondly, GREEDY achieves factor at least $\frac{5}{8}$ if the graph is planar.

Dyer, Frieze, and Pittel [DFP93] showed that on large sparse random graphs GREEDY has optimal expected approximation ratio; in particular, GREEDY computes (near) perfect matchings.

MRG. Goldschmidt and Hochbaum [GH90] analyzed the expected approximation ratio of MRG on large random graphs. For constant edge probability $0 < p < 1$ only very few nodes are not matched: in particular, at most $t_{|V|}$ nodes are not matched w.h.p. for an arbitrarily slowly growing function $t_{|V|}$ with $\lim_{|V| \rightarrow \infty} t_{|V|} = \infty$. Moreover, if the

¹¹As was pointed out in [DFP93], the proof of Tinhofer's other theorem, stating that there is a large interval for $1 - p$ such that MRG performs strictly better than GREEDY, is flawed.

¹²The *girth* of a graph is the minimum length of a contained cycle.

edge probability $\lim_{|V| \rightarrow \infty} p = 1$ grows arbitrarily slowly, i.e. if the number of edges is super-linear, then MRG produces a perfect matching w.h.p.

Another result of Dyer et al. [DFP93] also shows that the expected approximation ratio of MRG converges to 1 for large random graphs.

Aronson, Dyer, Frieze, and Suen [ADFS95] showed that MRG achieves expected approximation ratio at least $\frac{1}{2} + \frac{1}{400,000}$ on general graphs. While this bound might seem very small at first glance, it was the first—and for almost two decades remained the only—non-trivial bound on the expected approximation ratio of a greedy matching algorithm.

Aronson et al. also showed, similar to [DF91], a bound on MRG’s expected approximation ratio that depends on the density of a class of graphs. An implication of this result is that, as for GREEDY, the performance of MRG on graphs with degree at most Δ is at least $\frac{\Delta}{2\Delta-1}$.

SHUFFLE. The expected approximation ratio of the SHUFFLE algorithm was bounded by at least $\frac{2(5-\sqrt{7})}{9} \approx 0.523$ by Chan, Chen, Wu, and Zhao [CCWZ14]. For bipartite graphs Poloczek [Pol12] gave a bound of at least 0.696. Goel and Tripathi [GT12] and Poloczek [Pol12] showed a 0.727 inapproximability bound by utilizing a known bound for the classical RANKING algorithm in the bipartite ROA model (random order arrival).

MINGREEDY and Variants. For 3-regular random graphs Frieze, Radcliffe, and Suen [FRS95] showed that, in expectation, the MINGREEDY algorithm leaves only $\lambda_{|V|}$ nodes unmatched, where $c_1|V|^{1/5} \leq \lambda_{|V|} \leq c_2|V|^{1/5} \log |V|$ holds for constants c_1, c_2 . This result very accurately reflects their experimental observations in the same work, see Section 1.4.1.

MINGREEDY can be understood as a refinement of MRG, since the chance of picking an edge which belongs to a maximum matching (in the reduced graph) is higher for nodes of small degree. However, somehow surprisingly, MINGREEDY performs worse than MRG, as was shown by Poloczek [Pol12, BP15]: a family of bipartite graphs was constructed such that for any $\varepsilon > 0$ the MINGREEDY algorithm achieves expected approximation ratio at most $\frac{1}{2} + \varepsilon$ w.h.p.

The construction in [Pol12, BP15] also showed that algorithms MDS and DOUBLE-MINGREEDY are not an improvement over MRG or MINGREEDY, since a $\frac{1}{2} + \varepsilon$ -bound on their expected approximation ratios is established as well. The $\frac{1}{2}$ -bound for MDS also follows from a construction presented by Hougardy [Hou09]; a slight modification of Hougardy’s construction also implies the $\frac{1}{2}$ -bound for DOUBLE-MINGREEDY.

Shapira [Sha97] claimed (without proof) the existence of graphs for which the DOUBLE-MINGREEDY algorithm computes matchings of size “only about half the size of a maximum matching”. Shapira also showed that the solution of DOUBLE-MINGREEDY always contains at least $\min\left(\left\lfloor |V| + \frac{1}{2} - \sqrt{|V|^2 - |V| - 2|E| + \frac{9}{4}} \right\rfloor, \left\lfloor \frac{3}{4} + \sqrt{\frac{|E|}{2} - \frac{7}{16}} \right\rfloor\right)$ edges. This bound applies to the variant of the algorithm in which all ties are broken in worst-case fashion, i.e. it is a guarantee on the obtained matching size.

The KARSIPSER Algorithm. Karp and Sipser [KS81] proposed a matching algorithm based on node contractions. They analyzed a simplified version of this algorithm, which became known as the KARSIPSER algorithm. They showed that on large sparse random graphs, KARSIPSER asymptotically computes nearly optimal matchings. Aronson, Frieze, and Pittel [AFP98] improved error terms of the analysis in [KS81].

Inapproximability Results for Classes of Greedy-Like Algorithms. Rather than analyzing limits on the approximation performance of particular algorithms, further studies investigated how successful an algorithm can be given restrictions on the access to the input graph.

Vertex Iterative Algorithms. The $\frac{3}{4}$ -inapproximability bound for SHUFFLE on general graphs also applies to the class of so called *vertex iterative* algorithms [GT12]. We denote this class as \mathcal{VI} . A randomized \mathcal{VI} -algorithm considers one node at a time, and for the current node *probes* for neighbors. The first successful probe determines the node matched with the current node. The algorithm may also decide to stop probing for neighbors of the current node, thereby declaring that the current node will never be matched. The next node is chosen adaptively based on the outcomes of all previous computations.

Which algorithms belong to \mathcal{VI} ? The SHUFFLE algorithm probes for neighbors in the order determined by the permutation which is computed in the beginning. The MRG algorithm belongs to \mathcal{VI} as well, since it probes for neighbors in random order.

Algorithms for the Query Commit Problem. The class of algorithms for the *query commit* problem is denoted as \mathcal{QC} : unlike a \mathcal{VI} -algorithm, a \mathcal{QC} -algorithm does not have to consider nodes one after the other but may probe for edges in an arbitrary randomized fashion (e.g. the GREEDY algorithm belongs to \mathcal{QC}). Each found edge has to be picked. The pairwise kidney exchange problem, discussed in the beginning of Section 1, is an instantiation of the query commit problem, since after a successful probe the according

cross-transplant has to be executed. The class \mathcal{QC} contains all vertex iterative algorithms, i.e. we have

$$\mathcal{VI} \subseteq \mathcal{QC},$$

as was pointed out in [GT12]. In the same work, an inapproximability bound of 0.7916 for general graphs was shown for \mathcal{QC} -algorithms.

Fully Randomized Priority Algorithms. In [Pol12] it was shown that no algorithm in the class of *fully randomized priority algorithms in the node model* can approximate matching better than factor $\frac{5}{6}$, in expectation. This class contains all \mathcal{QC} -algorithms as well as MINGREEDY and the KARPSIPSER algorithm [BP15].

Hypergraph Matching. In a k -hypergraph an edge contains up to k nodes; in a k -uniform hypergraph each edge contains exactly k nodes. Note that graphs are included as the special case $k = 2$. In the k -Hypergraph Matching Problem (also known as the k -Set Packing Problem) the goal is to find a largest possible set of node disjoint edges. Unlike matching for common graphs, the k -Hypergraph Matching Problem is \mathcal{NP} -complete.¹³ A $\frac{1}{k}$ -approximation is easily obtained by greedily picking arbitrary edges [KH78].

Aronson et al. [ADFS95] adapted the GREEDY heuristic to k -uniform hypergraphs and showed that the expected approximation ratio is at least $1/(k - \frac{k-1}{m})$, where the non-negative value of m depends on the graph.

Bennett and Bohman [BB12] showed the following bound on the expected performance of GREEDY on k -uniform D -regular hypergraphs with N nodes: if $D \rightarrow \infty$ as $N \rightarrow \infty$ and co-degrees are at most $L = o(D/\log^5 N)$, then a proportion of at most $(L/D)^{\frac{1}{2(k-1)} + o(1)}$ nodes remains unmatched w.h.p.

Also for k -uniform hypergraph matching, local search was shown to allow for non-trivial approximation performance. Hurkens and Schrijver [HS89] gave, for any fixed $\varepsilon > 0$, a polynomial time local search algorithm with approximation ratio $\frac{k}{2} + \varepsilon$. Using an enhanced local search method, Cygan [Cyg13] improved the approximation ratio to $\frac{k+1+\varepsilon}{3}$. On the other hand, Hazan, Safra, and Schwartz [HSS06] showed that k -uniform hypergraph matching cannot efficiently be approximated within a factor of $O(\frac{k}{\ln k})$.

¹³The 3-dimensional matching problem, where each edge contains exactly three nodes and the graph is tripartite, as well as the unrestricted hypergraph matching problem belong to Karp's 21 \mathcal{NP} -complete problems. For an overview of problems closely related to hypergraph matching see [CL12].

1.4.3 Exact Algorithms

For completeness of this thesis we include a brief overview of exact polynomial time algorithms for matching. Also, the key concept of an *augmenting path*, which lies at the core of many exact algorithms, is fundamental to some of our main results.

Given a maximal but non-maximum matching M , Berge’s Lemma [Ber57] shows that there exists an *augmenting path* which can be used to “adjust” M and increase its size by one.¹⁴ This fact is used in the following classical algorithms, which grow an arbitrary initial matching by repeatedly finding augmenting paths: the algorithms of Edmonds [Edm65], of Hopcroft and Karp [HK73] (this one is for bipartite graphs) and of Micali and Vazirani [MV80, Vaz13, Gab14]. Berge’s Lemma also shows that the obtained matching is of maximum size as soon as no further augmenting path can be found.

Edmond’s algorithm was implemented in time $O(|V|^3)$ by Gabow [Gab76] and improved to time $O(|V| \cdot |E|)$ by Gabow and Tarjan [GT85].

Asymptotically faster implementations exist for the algorithms of Hopcroft and Karp and of Micali and Vazirani, and both algorithms are among the fastest exact algorithms in practice. In both algorithms, in each iteration the current matching is grown using a largest possible set of node-disjoint augmenting paths of currently smallest size. In particular, in each iteration the minimum length of an augmenting path increases, and it can be shown that after $O(\sqrt{|V|})$ iterations a maximum matching is obtained [HK73]. Each iteration can be performed in linear time $O(|E|)$, for both bipartite and non-bipartite graphs. Therefore both algorithms run in time $O(\sqrt{|V|} \cdot |E|)$.¹⁵

Another algorithm with the same asymptotic runtime is that of Gabow and Tarjan [GT91], which is a modification of their algorithm for minimum-cost matching on general graphs with integral edge weights.

On dense graphs, the above algorithms might take runtime as much as $O(|V|^{2.5})$. Also relying on augmenting path methods, this bound was improved by Alt, Blum, Mehlhorn, and Paul [ABMP91]: using certain adjacency matrix scanning techniques, their algorithm runs in time $O(|V|^{1.5} \sqrt{|E|/\log|V|})$ on bipartite graphs. Based on flow techniques, the algorithm of Goldberg and Karzanov [GK04] computes maximum matchings for

¹⁴To be concise we do not discuss *blossoms*, which are used to efficiently compute augmenting paths in non-bipartite graphs. See [Blu15] for an outline of the history of efficient computations of augmenting paths.

¹⁵A fast and straightforward approach to obtain an approximate matching is to iterate the algorithm of Micali and Vazirani until the initial matching was grown “large enough”: once all augmenting paths are longer than $2k + 1$, the obtained matching has size at least $\frac{k}{k+1}$ times optimal, as was shown in [HK73]. Since each iteration takes linear time, for any fixed $\varepsilon > 0$ a matching within $1 - \varepsilon$ of maximum size can be computed in linear time, where the constant in the asymptotic runtime depends on ε .

general graphs in time $O\left(\sqrt{|V|} \cdot |E| \cdot \frac{\log(|V|^2/|E|)}{\log|V|}\right)$. The randomized algorithm of Mucha and Sankowski [MS04] builds upon algebraic methods and runs in time $O(|V|^\omega)$ on general graphs, where $\omega < 2.38$ is the exponent of the fastest algorithm for multiplying two $|V| \times |V|$ -matrices; this algorithm was simplified by Harvey in [Har09].

2 Performance Guarantees: Basics

We develop the basics of our analysis of 1-2-GREEDY and the KARPSSIPER algorithm. Our approximation *guarantees* hold even under the assumption of worst case tie breaking.

In each step, either algorithm picks an arbitrary edge unless a node of degree at most two resp. of degree one exists, in which case an arbitrary edge incident with a node of minimum degree is picked. In our analysis, we consider the following equivalent reformulation of these algorithms. Since we assume worst case tie breaking, picking an arbitrary edge is equivalent with first selecting an arbitrary node, then an arbitrary neighbor, and picking the edge connecting both nodes. Therefore, throughout Sections 2 to 4 we analyze 1-2-GREEDY and KARPSSIPER as formulated in Algorithm 2. We implicitly refer to *current* (non-zero) degrees in the reduced graph unless explicitly stated otherwise. (Algorithm 2 is a refined version of Algorithm 1: Lines 3 to 8 implement the selection routine of Line 3 in Algorithm 1.)

Algorithm 2 Algorithms 1-2-GREEDY ($d = 2$) and KARPSSIPER ($d = 1$).

```
1.  $M \leftarrow \emptyset$ 
2. while  $E \neq \emptyset$  do
3.   if there is a node of degree at most  $d$  then
4.     let  $u$  be an arbitrary node of minimum degree
5.   else
6.     let  $u$  be an arbitrary node
7.   end if
8.   select neighbor  $v$  of  $u$ 
9.    $M \leftarrow M \cup \{u, v\}$ 
10.   $E \leftarrow E \setminus \{\{x, y\} : x \in \{u, v\}, y \in |V|\}$ 
11. end while
12. return  $M$ 
```

Simple Guarantees Using Degree Information. Our analysis must be tailored to algorithms whose edge selection routines utilize node degree information. Recall that algorithms GREEDY, MRG, and SHUFFLE (cf. List 1 on page 16) do not incorporate degree information to select the next edge. The approximation ratio of these algorithms—as well as of any other algorithm that does not utilize degree information—is exactly $\frac{1}{2}$, even in bipartite graphs of maximum degree two. Why? Since in the graph $\circ-\circ-\circ-\circ$ the middle edge might be picked first.

As is well known, picking an edge incident with a node of degree 1 is “optimal” in the following sense (see e.g. Corollary 3.1.6 in [LP86, p. 88]).

Proposition 1 (Folklore). *If node u in edge $\{u, v\}$ has degree one, then $\{u, v\}$ belongs to some maximum matching in the graph.*

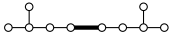
Proof. We construct a maximum matching which contains edge $\{u, v\}$. Assume that M^* is a maximum matching with $\{u, v\} \notin M^*$. Then v must be matched in M^* , since otherwise $\{u, v\}$ could be added to M^* , implying that M^* is not optimal. Say v is matched in edge $\{v, w\} \in M^*$. We replace $\{v, w\}$ with $\{u, v\}$ and obtain a matching of size $|M^*|$ which includes $\{u, v\}$. \square

Picking an edge incident with a node of degree 1 whenever possible allows to easily obtain maximum matchings in the following graphs.

Proposition 2 (Folklore). *If every node has degree at most two, then algorithms 1-2-GREEDY, DOUBLE-MINGREEDY, MDS, and the KARPSIPSER algorithm compute a maximum matching.*

Proof. The connected components of such graphs are paths and cycles. Edges of a path are always picked for a node of degree 1 at an end of the path—which is optimal. Once a cycle is “cracked open” it becomes a path and is then solved optimally. \square

Proposition 3 (Folklore). *Algorithms 1-2-GREEDY, DOUBLE-MINGREEDY, and KARP-SIPSER compute maximum matchings in forests.*

(For forests, MDS is not an optimal algorithm, as the following example shows. The bold edge in the graph  has minimum degree sum, since all edges have degree sum at least four. Hence the dotted edge might be picked first. Consequently, the approximation ratio of MDS is at most $\frac{3}{4}$.)

Proof. In each step an edge incident with a node of degree one is picked. \square

Organization of the Section. Since optimality of 1-2-GREEDY and the KARP-SIPSER algorithm is guaranteed if degrees are bounded by at most two (cf. Proposition 2), from here on we focus on graphs in which every degree is at most Δ for $\Delta \geq 3$.

By \mathcal{A} we denote 1-2-GREEDY resp. the KARP-SIPSER algorithm, and we discuss our approach to prove that algorithm \mathcal{A} achieves approximation ratio at least $\alpha > \frac{1}{2}$, where α will later be chosen appropriately.

In Section 2.1 we introduce the connected components of a certain graph $H(\mathcal{A})$ defined by \mathcal{A} . In Section 2.2 we present our technique to bound the “local” approximation ratio of each component of $H(\mathcal{A})$ in order to obtain a bound on the (global) approximation ratio of \mathcal{A} .

2.1 Alternating Paths and Cycles

Let $G = (V, E)$ be the input graph. W.l.o.g. we may assume that G is connected, since our performance guarantees apply to each connected component separately.

Let M^* be an arbitrary but fixed maximum matching in G . By M we denote the matching computed by \mathcal{A} on input G . In our analysis, we study the connected components of the graph

$$H(\mathcal{A}) = (V, M \cup M^*).$$

Nodes in $H(\mathcal{A})$ have degree at most two. Therefore any connected component is a sequence $\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{k-1}, v_k\}$ of edges. In the following we present the different types of sequences (in diagrams we depict an M^* -edge as $\text{○}=\text{○}$ and an M -edge as $\text{○}\#\text{○}$). In particular, we show that w.l.o.g. we may assume in our analysis that only two types of edge sequences exist, namely *singletons* and *paths*.

✓ *Singletons.*

A singleton is an edge contained in both M and M^* : $\text{○}\#\text{○}$.

• Alternating edge sequences.

An alternating edge sequence alternates between edges of M and M^* . We distinguish such sequences by their length.

✗ Alternating edge sequences of even length.

Note that $v_1 = v_k$ might hold, in which case the sequence is a cycle. For convenience, throughout our analysis we assume w.l.o.g. that even-length alternating edge sequences do not exist. We may do so, as we show next.

Lemma 4. *There exists a maximum matching M^* such that no component of $H(\mathcal{A})$ is an alternating edge sequence of even length.*

Proof. Let some maximum matching N^* be given. Assume that a component of the graph $(V, M \cup N^*)$ is an alternating edge sequence of even length, call that component X . We modify N^* appropriately without decreasing $|N^*|$ such that X is replaced by singletons only: e.g. $\text{○}=\text{○}\#\text{○}$ becomes $\text{○}\text{---}\text{○}\#\text{○}$, where an unmarked edge $\text{○}\text{---}\text{○}$ belongs neither to M nor to N^* . After repeating the process until there are no more even-length alternating edge sequences left, set $M^* = N^*$. \square

Steps for Endpoints. For each path X we also define the *first step for an endpoint of X* and the *second step for an endpoint of X* , see the crossed edges in Figure 2. The first step for an endpoint of X is the earliest step in which algorithm \mathcal{A} matches an M^* -neighbor of an endpoint w of X . In the second step for an endpoint of X , algorithm \mathcal{A} matches an M^* -neighbor of an endpoint w' of X with $w' \neq w$.

Observe that the first step for an endpoint of X might also create X . For a path with only one M -edge, the creation step and both steps for endpoints of X coincide.

2.2 Amortization

Based on the connected components of $H(\mathcal{A})$ we discuss our approach for the proof of performance guarantees for algorithm \mathcal{A} .

2.2.1 Local Approximation Ratios

Let X denote a path or singleton of $H(\mathcal{A})$. The numbers of M -edges and M^* -edges in component X are denoted by m_X resp. m_X^* . (In particular, note that we have $|M| = \sum_X m_X$ and $|M^*| = \sum_X m_X^*$.) To motivate our approach, the fraction

$$\frac{m_X}{m_X^*} \tag{1}$$

is of central interest. Observe that we have $\frac{1}{2} \leq \frac{m_X}{m_X^*} \leq 1$ for any component X . Why? First, for a shortest possible path X we have $m_X = 1$ and $m_X^* = 2$. Secondly, for longer paths (1) converges to one (recall that $m_X^* = m_X + 1$ holds). Lastly, for a singleton X we have $m_X = m_X^* = 1$.

In particular, for short paths (1) is smaller than α . This is especially true for each path X with only $m_X = 1$ edge of M , which we also call a $\frac{1}{2}$ -path.

Our goal is to develop a charging scheme that allows to amortize small fractions $\frac{m_X}{m_X^*}$ with large ones such that all fractions are at least α . Intuitively, we redistribute “funds” from “rich” to “poor” components without introducing new poverty. Therefore we move (equally valuable) coins and bills between components.

Definition 5 (Balance). *For node x in $H(\mathcal{A})$ we denote the number of coins received by x as rcv_x° and the number of coins payed by x as pay_x° . The number of bills received and payed by x is denoted as rcv_x^\square resp. pay_x^\square .*

Let $\triangle \in \{\circ, \square\}$. For groups of nodes we denote composite payments as follows. For edge $\{x, x'\}$ in $H(\mathcal{A})$ we let $\text{pay}_{\{x, x'\}}^\triangle = \text{pay}_x^\triangle + \text{pay}_{x'}^\triangle$. For any component X in the

graph $H(\mathcal{A})$ we let $\text{pay}_X^\triangleleft = \sum_x \text{pay}_x^\triangleleft$, where we sum over all nodes x which belong to component X . Analogously, we define $\text{rcv}_{\{x,x'\}}^\triangleleft$ for edge $\{x,x'\}$ and $\text{rcv}_X^\triangleleft$ for component X .

The balance of component X is defined as

$$\text{bal}_X = \text{rcv}_X^\circ + \text{rcv}_X^\square - \text{pay}_X^\circ - \text{pay}_X^\square.$$

No funds are wasted in the redistribution, i.e. summing over all components X we have

$$\sum_X \text{rcv}_X^\circ = \sum_X \text{pay}_X^\circ \quad \text{as well as} \quad \sum_X \text{rcv}_X^\square = \sum_X \text{pay}_X^\square. \quad (2)$$

Let X be a component. Using the amount bal_X of funds owned by X , we adjust the portion of $|M|$ in the fraction $\frac{m_X}{m_X^*}$ as follows (recall that $|M| = \sum_X m_X$ holds).

Definition 6 (Local Approximation Ratio). *Let the coin and bill value $\kappa > 0$ be given. The local approximation ratio of component X is*

$$\frac{m_X + \kappa \cdot \text{bal}_X}{m_X^*} = \frac{m_X + \kappa \cdot \text{rcv}_X^\circ + \kappa \cdot \text{rcv}_X^\square - \kappa \cdot \text{pay}_X^\circ - \kappa \cdot \text{pay}_X^\square}{m_X^*}.$$

If the balance bal_X is large enough for $\frac{m_X + \kappa \cdot \text{bal}_X}{m_X^*} \geq \alpha$ to hold, then X is called α -balanced.

Since the local approximation ratio of a component X grows with the balance of X , to obtain lower bounds on local approximation ratios we have to come up with appropriate lower bounds on the balance of singletons and paths. We allow negative balances for singletons and long augmenting paths, since these components have large fractions $\frac{m_X}{m_X^*}$, i.e. they are rich enough to pay funds without harm.

If all components are α -balanced, then our proof is complete, as we show next.

Lemma 7. *If each component X of $H(\mathcal{A})$ is α -balanced, then algorithm \mathcal{A} achieves (global) approximation ratio at least α .*

Proof. Rewriting (2) as $\sum_X (\text{rcv}_X^\circ - \text{pay}_X^\circ + \text{rcv}_X^\square - \text{pay}_X^\square) = \sum_X \text{bal}_X = 0$, we obtain

$$\frac{|M|}{|M^*|} = \frac{\sum_X m_X}{\sum_X m_X^*} = \frac{\sum_X m_X + \kappa \cdot \text{bal}_X}{\sum_X m_X^*} \geq \frac{\sum_X \alpha \cdot m_X^*}{\sum_X m_X^*} = \alpha. \quad \square$$

2.2.2 Transfers

To redistribute funds, we utilize edges of the input graph G . In particular, our charging scheme has to reflect that algorithm \mathcal{A} selects a node v because of its degree: funds are moved over edges being incident with v when v gets matched.

However, the M -edge or M^* -edge of v connects v with another node in the same component and hence does not allow us to redistribute funds. Therefore, we utilize edges which connect different components of $H(\mathcal{A})$, namely edges in the set

$$F = E \setminus (M \cup M^*).$$

Are F -edges guaranteed to be incident with nodes of poor paths? In Lemma 9 a) we show that each endpoint of a path is incident with at least one F -edge. We prepare Lemma 9 in Lemma 8.

Lemma 8. a) Assume that algorithm \mathcal{A} creates path X in step t , say when selecting node u with degree $d_t(u)$ in the reduced graph. Then we have $d_t(u) \geq 2$.

b) Let w be an endpoint of a path in $H(\mathcal{A})$. The degree of w in the input graph G is $d_1(w) \geq 2$.

Proof. We prove a). Let X be a path. In the creation step t of X all M -edges and M^* -edges of X still belong to the reduced graph G_t . Consequently, the nodes of M -edges in X have degree at least two. In particular, the node selected in step t by \mathcal{A} has degree at least two.

We prove b). Assume that w is endpoint of path X and that X is created in step t . First, observe that the degree $d_1(w)$ of w in the input graph is at least $d_1(w) \geq d_t(w)$, i.e. at least as large as the degree of w in step t , since algorithm \mathcal{A} does not add edges into the graph.

To prove that $d_1(w) \geq 2$ holds we show that we have $d_t(w) \geq 2$. Consider the creation step t of path X and let v be an arbitrary non-isolated node. By a) and by definition of algorithm \mathcal{A} , node v has degree at least $d_t(v) \geq 2$. It suffices to show that endpoint w is not isolated in step t . To see this, observe that w is incident with its M^* -edge when X is created in step t . \square

Lemma 9. a) Let w be an endpoint of path X in $H(\mathcal{A})$. In the input graph G , endpoint w is incident with at least one F -edge.

b) If $\{w, u\}$ is an F -edge incident with endpoint w , then u is an M -matched node.

Proof. Statement a) is a direct consequence of Lemma 8 b), since exactly one of at least two edges incident with w in the input graph is *not* an F -edge, namely the M^* -edge of w .

To prove b), observe that if both w and u are endpoints, then $M \cup \{\{u, w\}\}$ is a matching, contradicting the fact that M is a maximal matching. \square

Note that moving funds over F -edges incident with path endpoints is natural in the sense that such F -edges are the only F -edges guaranteed to exist, as the example in Figure 3 shows.

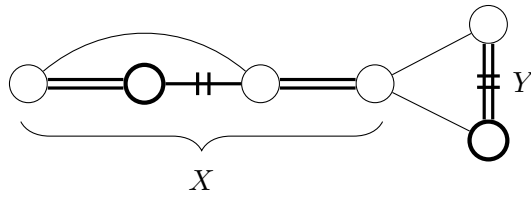


Figure 3: A $\frac{1}{2}$ -path X and a singleton Y connected only by F -edges incident with path endpoints (the algorithm selects bold nodes from left to right)

Transfer Edges. We do not move funds over each F -edge incident with a path endpoint, but we carefully choose only some of them. In the following definition we define *transfer* edges: each transfer moves exactly one coin, but no bills.¹⁶ Thereafter, in Lemmas 12 and 13 we bound the number of coins payed and received by nodes. Eventually, in a summary we combine our results to obtain basic bounds on the number of coins owned by singletons and paths.

Definition 10 (Transfer). *Let edge $\{v, w\}$ be an edge in the set $F = E \setminus (M \cup M^*)$, where node v is M -matched and w is a path endpoint. Assume that v becomes matched in step t (when $\{v, w\}$ is removed from the reduced graph G_t).*

If, after $\{v, w\}$ is removed, the degree of w is at most $d_{t+1}(w) \leq 1$, then $\{v, w\}$ is called a transfer and one coin is moved from the component of v to the component of w .

We denote a transfer edge $\{v, w\}$ as (v, w) .

¹⁶Later we also introduce so-called *donations*. Donations are F -edges as well, but they move only bills instead of coins. If an F -edge is a donation then it is not a transfer, and vice versa, i.e. the set of transfers edges and the set of donation edges are disjoint.

Bounds On Coin Payments. How many coins does a given node v have to pay? Since a transfer moves exactly one coin and a transfer is an F -edge, the number of coins paid by v is bounded by the number of F -edges incident with v . Therefore we determine bounds as functions of Δ in Lemma 12. On the other hand, the number of coins received by an endpoint w is *not* a function of Δ , since transfers only point to certain “small degree” endpoints. First, we introduce necessary notation in the following definition.

Definition 11 (Bounds on Payments).

- Let x be an M -matched node in component X , and $\{x, x'\}$ the M -edge of x . By PAY_x° we denote an upper bound on the number of coins paid by node x :

$$\text{pay}_x^\circ \leq \text{PAY}_x^\circ.$$

The maximum number of coins paid by

- M -edge $\{x, x'\}$ is defined as $\text{PAY}_{\{x, x'\}}^\circ = \text{PAY}_x^\circ + \text{PAY}_{x'}^\circ$.
- component X is defined as $\text{PAY}_X^\circ = \sum_x \text{PAY}_x^\circ$, where we sum over all M -matched nodes x in X .
- Let w and w' be the endpoints of path Y . By RCV_w° we denote a lower bound on the number of coins received by w , i.e. we have

$$\text{rcv}_w^\circ \geq \text{RCV}_w^\circ.$$

Now define $\text{RCV}_Y^\circ = \text{RCV}_w^\circ + \text{RCV}_{w'}^\circ$. (Note that a definition of RCV_e° for an edge e is unnecessary, since there is no path edge which connects two path endpoints.)

- By BAL_X we denote a bound on the balance $\text{bal}_X = (\text{rcv}_X^\circ - \text{pay}_X^\circ) + (\text{rcv}_X^\square - \text{pay}_X^\square)$ of component X , i.e. we have

$$\text{bal}_X \geq \text{BAL}_X.$$

Lemma 12. Let x be an M -matched node in component X , and $\{x, x'\}$ the M -edge of x .

a) If X is a singleton, then

$$\text{PAY}_x^\circ = \Delta - 1$$

is an upper bound for pay_x° and we have $\text{PAY}_{\{x, x'\}}^\circ = \text{PAY}_X^\circ = 2 \cdot (\Delta - 1)$.

b) If X is a path, then

$$\text{PAY}_x^\circ = \Delta - 2$$

is an upper bound for pay_x° . Also, we have $\text{PAY}_{\{x, x'\}}^\circ = 2 \cdot (\Delta - 2)$ as well as $\text{PAY}_X^\circ = m_X \cdot 2(\Delta - 2)$.

Proof. Note that in a) as well as b) we only need to verify that $\text{pay}_x^\circ \leq \text{PAY}_x^\circ$ holds for node x , since $\text{PAY}_{\{x,x'\}}^\circ$ for edge $\{x,x'\}$ and PAY_X° for component X follow by Definition 11.

We bound the number of transfers leaving x . Each transfer is an F -edge in the input graph G , hence it suffices to bound the number of F -edges incident with x in G .

a) If x belongs to a singleton, then x is incident with at most $\Delta - 1$ many F -edges, since x is also incident with its M -edge.

b) If x belongs to a path, then x is incident with at most $\Delta - 2$ many F -edges, since x is also incident with its M -edge and its M^* -edge. \square

Lemma 13. *Let w be an endpoint of path X . Then*

$$\text{RCV}_w^\circ = 1$$

and $\text{RCV}_X^\circ = 2$ are lower bounds on rcv_w° and rcv_X° , respectively.

Proof. We only need to verify that $\text{rcv}_w^\circ \geq \text{RCV}_w^\circ$ holds for endpoint w , since RCV_X° for path X follows by Definition 11. We show that w receives at least one transfer.

Since w is a path endpoint, node w never gets matched. By Lemma 8, the degree of w in the input graph is at least $d_1(w) \geq 2$. Hence, algorithm \mathcal{A} removes the M^* -edge of w and the $d_1(w) - 1 \geq 1$ many F -edges of w .

Assume that in step t the degree of w drops from $d_t(w)$ to $d_{t+1}(w) = 0$, and observe that we have $d_t(w) = 1$ or $d_t(w) = 2$. In either case we show that w receives a coin.

First, assume that $d_t(w) = 2$ holds. Two edges incident with w are removed from the (reduced) graph in step t . Observe that at least one of the removed edges must be an F -edge, call it e . By Definition 10 (**Transfer**), edge e is a transfer and one coin is moved to w along e .

Now assume that $d_t(w) = 1$ holds. The last edge incident with w , call it e , is either an F -edge or an M^* -edge. If e is an F -edge, then e is a transfer and one coin is moved to w along e . If e is an M^* -edge. Then in an earlier step $t' < t$ the degree of w dropped to $d_{t'+1}(w) = 1$ when an F -edge e' incident with w was removed. Edge e' is a transfer along which one coin is moved to w . \square

Summary. We summarize Lemmas 12 and 13 to obtain basic bounds on the number of coins owned by singletons and paths.

- A singleton X receives no transfers, since only path endpoints receive transfers. Therefore we define $\text{rcv}_X^\circ = 0 = \text{RCV}_X^\circ$. By Lemma 12 a), singleton X pays at most $\text{pay}_X^\circ \leq \text{PAY}_X^\circ = 2(\Delta - 1)$ coins. Hence X owns at least

$$\begin{aligned} \text{rcv}_X^\circ - \text{pay}_X^\circ &\geq \text{RCV}_X^\circ - \text{PAY}_X^\circ \\ &\stackrel{\text{RCV}_X^\circ=0}{=} -2(\Delta - 1) \end{aligned} \quad (3)$$

coins.

- A path X pays at most $\text{pay}_X^\circ \leq \text{PAY}_X^\circ = 2m_X(\Delta - 2)$ coins by Lemma 12 b). Moreover, path X receives at least $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ = 2$ coins by Lemma 13. Therefore X owns at least

$$\begin{aligned} \text{rcv}_X^\circ - \text{pay}_X^\circ &\geq \text{RCV}_X^\circ - \text{PAY}_X^\circ \\ &= 2 - 2m_X(\Delta - 2) \end{aligned} \quad (4)$$

coins, see Figure 4 for an illustration.

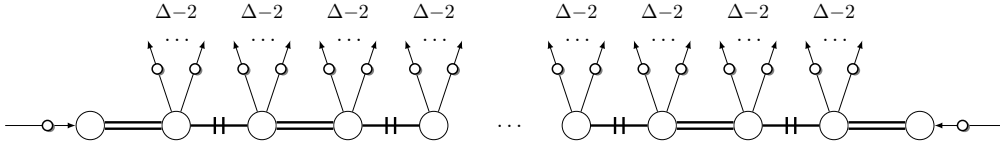


Figure 4: A path X owns at least $2 - 2m_X(\Delta - 2)$ coins (transfers are drawn as directed edges)

2.3 Isolated Nodes

We conclude the presentation of the basics and discuss our claim in Section 2.1 that we may conveniently ignore isolated $H(\mathcal{A})$ -nodes in the argument. We first give some intuition and thereafter provide a formal description.

Implicitly, isolated nodes *do* take part in the system of moving M -funds. In particular, we will see that an isolated node v might receive funds, but v does *not* pay funds. Consequently, node v effectively “removes” funds from the system and hence *does not* illegally increase the local approximation ratio of any other component in $H(\mathcal{A})$.

We formalize why we may go without explicitly considering isolated nodes. Recall that Definition 10 (**Transfer**) is formulated so as to focus attention to a path endpoint w receiving a coin.

Throughout the analysis, the only crucial property of path endpoint w is that w is *not M -matched*.

Now observe that generalizing Definition 10 from w being a path endpoint to w being a non- M -matched node includes isolated nodes in the system of moving funds, since these are non- M -matched as well (there are no further types of non- M -matched nodes).

Why does a non- M -matched node w pay no funds? By Definition 10, node w pays no transfers. The same holds for our second type of funds-moving F -edges, called *donations*, which we introduce in Definitions 26 and 41.

3 Performance Guarantees for 1-2-GREEDY

In this section we present our $\frac{\Delta-1}{2\Delta-3}$ performance guarantees for the 1-2-GREEDY algorithm on general degree bounded graphs. Recall that it suffices to show that all components in the graph $H(1-2-GREEDY)$ are $\frac{\Delta-1}{2\Delta-3}$ -balanced, i.e. that each component has local approximation ratio at least $\frac{\Delta-1}{2\Delta-3}$. Therefore, our task is to establish appropriate bounds on the balance of each component, cf. Definition 6 (Local Approximation Ratio).

We present appropriate bounds for singletons and paths in Lemma 14 below. Given Lemma 14, it remains to verify both balance bounds.

Lemma 14. *Let X be a component in the graph $H(1-2-GREEDY)$. Assume that the balance bal_X of X is at least*

$$\text{BAL}_X = -2(\Delta - 2) \quad \text{if } X \text{ is a singleton and} \quad (5 \text{ Singleton})$$

$$\text{BAL}_X = 2(\Delta - 1) - 2m_X(\Delta - 2) \quad \text{if } X \text{ is a path.} \quad (6 \text{ Path})$$

Then there exists a coin and bill value κ such that X and all other components of $H(1-2-GREEDY)$ are $\frac{\Delta-1}{2\Delta-3}$ -balanced.

Proof. Choose $\kappa = \frac{1}{2(2\Delta-3)}$. Using (5 Singleton), the local approximation ratio of a singleton X is at least $\frac{1+\kappa \cdot \text{bal}_X}{1} \geq 1 + \kappa \cdot \text{BAL}_X = 1 - 2\kappa(\Delta - 2)$, which in turn equals

$$1 - 2\kappa(\Delta - 2) = 1 - \frac{2(\Delta - 2)}{2(2\Delta - 3)} = \frac{2(2\Delta - 3) - 2(\Delta - 2)}{2(2\Delta - 3)} = \frac{\Delta - 1}{2\Delta - 3}. \quad (7)$$

Therefore each singleton is $\frac{\Delta-1}{2\Delta-3}$ -balanced.

Using (6 Path), the local approximation ratio of a path X is at least

$$\begin{aligned} \frac{m_X + \kappa \cdot \text{bal}_X}{m_X^*} &\geq \frac{m_X + \kappa \cdot \text{BAL}_X}{m_X^*} \\ &= \frac{m_X - 2\kappa m_X(\Delta - 2) + 2\kappa(\Delta - 1)}{m_X + 1} \\ &= \frac{m_X \cdot (1 - 2\kappa(\Delta - 2)) + 2\kappa(\Delta - 1)}{m_X + 1} \\ &= 1 - 2\kappa(\Delta - 2) + \frac{2\kappa(\Delta - 1) - (1 - 2\kappa(\Delta - 2))}{m_X + 1} \\ &= 1 - 2\kappa(\Delta - 2) + \frac{2\kappa(2\Delta - 3) - 1}{m_X + 1} \\ &= 1 - 2\kappa(\Delta - 2), \end{aligned}$$

which by (7) is bounded by at least $\frac{\Delta-1}{2\Delta-3}$ as well, i.e. all paths are $\frac{\Delta-1}{2\Delta-3}$ -balanced. \square

Organization of the Proof of (5 Singleton) and (6 Path). We analyze graphs of maximum degree $\Delta = 3$ in Section 3.1, where our proof relies only on coins moved in transfers—no bills are used. In Section 3.2 we analyze graphs of maximum degree $\Delta \geq 4$, where we also move bills.

3.1 Maximum Degree $\Delta = 3$

This section contains the proof of

Theorem 15. *The 1-2-GREEDY algorithm achieves approximation ratio at least $\frac{\Delta-1}{2\Delta-3} = \frac{2}{3}$ for graphs of maximum degree $\Delta = 3$.*

By Lemma 14, in the proof it suffices to verify balance bounds (5 Singleton) and (6 Path), which for $\Delta = 3$ reduce to

$$\text{BAL}_X = -2 \quad \text{if } X \text{ is a singleton and} \quad (8 \text{ Singleton}_{\Delta=3})$$

$$\text{BAL}_X = 4 - 2m_X \quad \text{if } X \text{ is a path.} \quad (9 \text{ Path}_{\Delta=3})$$

No Bills. We establish a proof of (8 Singleton $_{\Delta=3}$) and (9 Path $_{\Delta=3}$) that relies only on coins moved in transfers, i.e. we do not move bills and we do not utilize F -edges other than transfers to move funds. In particular, we assume that

$$\begin{aligned} \text{pay}_x^{\square} = \text{rcv}_x^{\square} = 0 & \quad \text{holds for any node } x, \text{ which implies} \\ \text{pay}_{\{x,x'\}}^{\square} = \text{rcv}_{\{x,x'\}}^{\square} = 0 & \quad \text{for any edge } \{x,x'\} \text{ and} \\ \text{pay}_X^{\square} = \text{rcv}_X^{\square} = 0 & \quad \text{for any component } X. \end{aligned}$$

Consequently, the balance of a component X is now simplified to owning coins:

$$\begin{aligned} \text{bal}_X &= \text{rcv}_X^{\circ} + \text{rcv}_X^{\square} - \text{pay}_X^{\circ} - \text{rcv}_X^{\square} \\ &= \text{rcv}_X^{\circ} - \text{pay}_X^{\circ}. \end{aligned}$$

Using this simplified definition of the balance, we introduce a more instructive formulation of bounds (8 Singleton $_{\Delta=3}$) and (9 Path $_{\Delta=3}$). Therefore we utilize results established after Definition 10 (Transfer), namely the lower bounds (3) and (4) on the number $\text{rcv}_X^{\circ} - \text{pay}_X^{\circ}$ of coins owned by a component X , see page 43. Recall that we denote the maximum number of coins paid by component X as PAY_X° and the minimum

number of coins received by path X as RCV_X° . Since $\text{bal}_X = \text{rcv}_X^\circ - \text{pay}_X^\circ$ holds, the balance of component X is at least

$$\begin{aligned} \text{bal}_X &\stackrel{(3)}{\geq} -\text{PAY}_X^\circ \stackrel{(3)}{=} -2(\Delta-1) = -4 && \text{if } X \text{ is a singleton and} \\ \text{bal}_X &\stackrel{(4)}{\geq} \text{RCV}_X^\circ - \text{PAY}_X^\circ \stackrel{(4)}{=} 2 - 2m_X(\Delta-2) = 2 - 2m_X && \text{if } X \text{ is a path.} \end{aligned}$$

Using these lower bounds, our reformulation of (8 [Singleton \$_{\Delta=3}\$](#)) and (9 [Path \$_{\Delta=3}\$](#)) is as follows. We demand that the balance bal_X of component X is at least

$$\begin{aligned} \text{BAL}_X = -2 &= -\text{PAY}_X^\circ + 2 && \text{if } X \text{ is a singleton and} \\ \text{BAL}_X = 4 - 2m_X &= \text{RCV}_X^\circ - \text{PAY}_X^\circ + 2 && \text{if } X \text{ is a path.} \end{aligned}$$

In particular, observe that we have to show that in our charging scheme the balance of each singleton or path is increased by at least two above the known lower bound (3) resp. (4).

3.1.1 Saving Coins

Recall that a singleton X only pays but does not receive coins. Our plan to verify (8 [Singleton \$_{\Delta=3}\$](#)) is to show that the nodes of X *save* two coins:

Definition 16 (Saving Coins). *Recall that an M -matched node x pays no more than PAY_x° coins. If x pays at most $\text{pay}_x^\circ \leq \text{PAY}_x^\circ - k$ coins, for $k > 0$, then we say that node x saves k coins. We say that component X saves K coins (denoted as $\text{pay}_X^\circ \leq \text{PAY}_X^\circ - K$) if the savings of all nodes of X sum up to at least K .*

So, if singleton X saves two coins, then $\text{bal}_X = -\text{pay}_X^\circ \geq -\text{PAY}_X^\circ + 2 = \text{BAL}_X$ holds, as desired.

To show that node x saves a coin, we study x in the step when x becomes matched. In particular, we study the F -edges which are incident with x in that step. We frequently apply the following arguments, which are direct consequences of Definition 10 ([Transfer](#)). For the sake of clarity, we formally verify each argument.

Lemma 17. *Assume that node x becomes M -matched in step t .*

- a) *Node x saves p coins if the degree of x in the input graph is $d_1(x) = \Delta - p$.*
- b) *Let f_i denote the number of F -edges incident with x in the reduced graph G_i . Node x saves q coins if we have $f_1 - f_t = q$, i.e. if q many F -edges were removed from x before x becomes matched.*

Assume that F -edges $\{x, y_1\}, \dots, \{x, y_k\}$ are incident with x in G_t .

- c) Node x saves r coins if nodes y_{i_1}, \dots, y_{i_r} ($i_j \neq i_{j'}$ for $j \neq j'$) are M -matched.
- d) Node x saves s coins if nodes y_{i_1}, \dots, y_{i_s} ($i_j \neq i_{j'}$ for $j \neq j'$) are endpoints but their degrees are at least $d_{t+1}(y_{i_j}) \geq 2$ after x becomes matched (for $1 \leq j \leq s$).
- e) In total, node x saves $p + q + r + s$ coins.

Proof. a) In the input graph all M -edges and M^* -edges of x are still incident with x . Hence p many F -edges are “missing”. For each such missing F -edge node x saves one coin.

b) By Definition 10 (Transfer), an F -edge can only be a transfer leaving x if it is incident with x when x becomes matched in step t . Hence for each F -edge removed before step t node x saves one coin.

c) By Definition 10 (Transfer), one node of a transfer is an endpoint. Hence for each F edge $\{x, y_{i_{j'}}\}$ for $1 \leq j' \leq r$ node x saves one coin.

d) By Definition 10 (Transfer), an F -edge $\{x, y_{i_{j'}}\}$ incident with x can only be a transfer if endpoint $y_{i_{j'}}$ has degree at most $d_{t+1}(y_{i_{j'}}) \leq 1$ after x becomes matched. Hence for each F edge $\{x, y_{i_{j'}}\}$ for $1 \leq j' \leq s$ node x saves one coin.

e) The p saved coins in a) are for “ F -edges” which are not contained in the input graph in the first place. The q saved coins in b) are for F -edges which are contained in the input graph but are removed before x becomes matched in step t . In c) and d) coins are saved for F -edges still being incident with x in step t . However, in c) an F -edge connects x with an M -matched node whereas in d) an F -edge connects x with a path endpoint. \square

For paths we proceed similar as for singletons. The balance of a path X is never smaller than $\text{bal}_X \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ$ by (4), whereas (9 Path $_{\Delta=3}$) requires $\text{bal}_X \geq \text{BAL}_X = \text{RCV}_X^\circ - \text{PAY}_X^\circ + 2$. If the nodes of X save two coins, then X is balanced since we have $\text{bal}_X = \text{rcv}_X^\circ - \text{pay}_X^\circ \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ + 2 = \text{BAL}_X$.

However, in the proof of (9 Path $_{\Delta=3}$) we have to deal with the case that path X saves *less* than two coins. In this case we show that path X receives sufficiently many *additional* coins.

Organization of the Proof. We start with the proof of two saved coins for singletons in Lemma 18 c). This concludes the proof of (8 Singleton $_{\Delta=3}$).

More generally, Lemma 18 also applies to paths and identifies those cases in which paths save two coins. Lemma 18 also prepares the analysis of the case in which a path X saves less than two coins. We argue in Lemma 19 that X saves at least one coin. Thus

if X saves less than two coins then we have $\text{pay}_X^\circ = \text{PAY}_X^\circ - 1$. If X saves only one coin, then Lemma 20 shows that X receives an additional coin, i.e. we have $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ + 1$. Hence by Lemmas 18 and 20 we have $\text{bal}_X = \text{rcv}_X^\circ - \text{pay}_X^\circ \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ + 2 = \text{BAL}_X$. This concludes the proof of (9 Path $_{\Delta=3}$).

Coins Saved in Creation Steps. Recall that a component X is created in the step when 1-2-GREEDY picks an M -edge of X for the first time. In the next result, we study situations in which coins are saved by a node becoming matched in the creation step of X . Intuitively, the argument proceeds as follows.

If 1-2-GREEDY creates X when selecting a node u of degree one, then recall that this step is “optimal”.¹⁷ Hence u should save many coins. Indeed, no F -edges are incident with u when u is selected. Therefore u has no incident transfers and pays no coins.

On the other hand, if node u has large degree $\Delta = 3$ when u is selected, then the next step of 1-2-GREEDY makes sure that u saves a coin. Why? Assume that u pays a coin over an incident transfer (u, w) . Then by Definition 10 (Transfer) endpoint w has degree one in the next step. Since the new minimum degree is one, 1-2-GREEDY selects a node y whose degree also dropped to one. Edge $\{u, y\}$ is not a transfer, since both y and u are M -matched. Therefore u saves a coin. By an analogous argument, the node matched with u saves an additional coin.

We also show that if X is a singleton, then X saves two coins, no matter what the degree of the node selected to create X is.

Lemma 18. *Assume that degrees are bounded by at most $\Delta = 3$. Consider the creation step t of component X , and assume that the 1-2-GREEDY algorithm selects node u with degree $d_t(u)$. Component X saves two coins*

- a) *if we have $d_t(u) = 1$, or*
- b) *if $d_t(u) = \Delta = 3$ holds, or*
- c) *if component X is a singleton.*

Proof. We prepare the argument. Assume that u becomes matched with v in step t . Recall that by Lemma 12 a) and b) the maximum number of coins paid by u is $\text{PAY}_u^\circ = \Delta - 1 = 2$ if X is a singleton resp. $\text{PAY}_u^\circ = \Delta - 2 = 1$ if X is a path. In particular, in both cases node u might have to pay a coin. Analogously, node v might have to pay a coin. Now observe that if neither u nor v pays a coin, then two coins are saved in total. In this case we are done.

¹⁷The edge incident with a node of degree one belongs to *some* optimal matching in the reduced graph. Note, however, that this optimal matching might be different from M^* .

So for the rest of the proof we assume that at least one of u and v pays a coin, say over transfer (x, w) for $x \in \{u, v\}$.

We prove **a)**. Recall the following: if in a step before creation of component X an edge $\{u, y\}$ incident with node u is removed from the graph, then $\{u, y\}$ must be an F -edge. Since the degree of u is $d_t(u) = 1$ when X is created, we obtain that $d_1(u) - d_t(u)$ many F -edges are removed from u before creation of X . Hence by Lemma **17 b)** node u saves $d_1(u) - d_t(u)$ coins. Moreover, by Lemma **17 a)** node u saves $\Delta - d_1(u)$ coins. By Lemma **17 e)**, we may sum both numbers. Therefore, in total, node u saves $(\Delta - d_1(u)) + (d_1(u) - d_t(u)) = 2$ coins.

We prove **b)**. Assume that $d_t(u) = 3$ holds. Since (x, w) is a transfer, in step t edge $\{x, w\}$ is removed from G_t and after step t the degree of w is $d_{t+1}(w) \leq 1$. Observe that in step t at most two edges incident with w are removed from the graph, since two nodes become matched. Since the degree of w is at least $d_t(w) \geq d_t(u) = 3$ in G_t , we have $d_{t+1}(w) \geq 1$ in step $t+1$. Using $d_{t+1}(w) \leq 1$ and $d_{t+1}(w) \geq 1$ we obtain $d_{t+1}(w) = 1$. But since endpoint w never becomes matched, another node $y \neq w$ with degree $d_{t+1}(y) = 1$ is selected next in step $t+1$ and becomes M -matched then. But node y had degree at least $d_t(y) \geq d_t(u) = 3$ in the creation step t of X . Hence in step t the degree of y drops from $d_t(y) = 3$ to $d_{t+1}(y) = 1$, namely when incident edges $\{u, y\}$ and $\{v, y\}$ are removed from G_t .

- **If y belongs to a component $Y \neq X$ other than X** , then both $\{u, y\}$ and $\{v, y\}$ are F -edges. But edges $\{u, y\}$ and $\{v, y\}$ are not transfers, since nodes u, v , and y are M -matched. Hence each of u and v saves a coin.
- **Assume that y is a node of X** . (Note that in this case X must be a path.) Nodes u, v , and y form a triangle in G_t and hence one of edges $\{u, y\}$ and $\{v, y\}$ is an F -edge, say edge $\{u, y\}$. But both u and y are M -matched, thus edge $\{u, y\}$ is not a transfer. Again, each of u and y saves a coin.

We prove **c)**. In this case X is a singleton. We may assume that $d_t(u) = 2$ holds in the creation step X , since **a)** and **b)** apply in particular to singletons. Either the degree of u in the input graph is $d_1(u) = 2$, i.e. it did not drop before step t . Or the degree of u is $d_1(u) = 3$ in the input graph, in which case we have $d_1(u) - d_t(u) = 1$. We apply Lemma **17 a)** resp. Lemma **17 b)** and obtain that u saves one coin.

Recall that we are done if X saves another coin besides the one saved by u . In order to obtain a contradiction, we assume that u saves the only coin. Recall that for singleton X

each of nodes u and v pays at most $\text{PAY}_u^\circ = \text{PAY}_v^\circ = 2$ coins. By our assumption, node u pays exactly one coin, say to w_u , and v pays exactly two coins, say to w_v and w'_v .

Recall that we assume that in the creation step of X node u has degree $d_t(u) = 2$. Therefore node u is adjacent to at most one of w_v and w'_v , since u is also adjacent to v . Since edges $\{u, w_u\}$, $\{v, w_v\}$, and $\{v, w'_v\}$ are transfers, Definition 10 (**Transfer**) implies that the degrees of w_u , w_v and w'_v are at most 1 after creation of X , i.e. in step $t + 1$. We distinguish the following two cases in step t .

- **If node u is adjacent to neither w_v nor w'_v in step t** , then endpoints w_u , w_v , and w'_v have degree exactly $d_{t+1}(w_u) = d_{t+1}(w_v) = d_{t+1}(w'_v) = 1$ after step t . Why? Their degrees are at least the minimum degree of $d_t(u) = 2$ in step t when $\{u, v\}$ is picked, and their degrees drop by exactly one in step t .
- Now consider the case that **u is adjacent to w_v or w'_v in step t** , say $w_u = w_v$ holds. Then the degree of w'_v drops by at most one when edge $\{u, v\}$ is picked. Since we have $d_t(w'_v) \geq d_t(u) = 2$, endpoint w'_v has degree exactly $d_{t+1}(w'_v) = 1$ afterwards.

In both cases, no other degrees than those of w_u , w_v , and w'_v drop in step t . Since in step $t + 1$ one of these endpoints has degree exactly one, one of w_u , w_v , or w'_v has the new minimum degree and becomes matched next. A contradiction, since path endpoints are never matched by 1-2-GREEDY. \square

3.1.2 Amortization for Paths

To finish the proof of Theorem 15 it remains to verify (9 **Path $_{\Delta=3}$**) for any given path X . Recall that the minimum balance of X is at least $\text{bal}_X = \text{rcv}_X^\circ - \text{pay}_X^\circ \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ$. But for (9 **Path $_{\Delta=3}$**) we have to verify a balance of at least

$$\text{bal}_X = \text{rcv}_X^\circ - \text{pay}_X^\circ \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ + 2.$$

We have already discussed that it suffices to show that X saves two coins, i.e. that $\text{pay}_X^\circ \leq \text{PAY}_X^\circ - 2$ holds. If X saves two coins then we are done. Consequently, in the rest of the proof it remains to consider the case that X saves less than two coins.

In Lemma 19 below we prove that each path saves at least one coin. In Lemma 20 we show that if a path X saves only one coin, i.e. if $\text{pay}_X^\circ = \text{PAY}_X^\circ - 1$ holds, then X receives at least $\text{rcv}_X^\circ \geq 3 = \text{RCV}_X^\circ + 1$ coins. Hence the balance of X is at least $\text{bal}_X = \text{rcv}_X^\circ - \text{pay}_X^\circ \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ + 2$, as required by (9 **Path $_{\Delta=3}$**).

Lemma 19. *Let X be a path which saves less than two coins. In the creation step t of X the 1-2-GREEDY algorithm selects a node u of degree $d_t(u) = 2$. Node u saves a coin for X and we have $\text{pay}_X^\circ = \text{PAY}_X^\circ - 1$.*

Proof. Recall that by Lemma 8 a) step t selects a node u of degree at least $d_t(u) \geq 2$. We analyze the cases that $d_t(u) = 2$ or $d_t(u) = 3 = \Delta$ holds. We obtain a contradiction in the latter case, since path X saves two coins by Lemma 18 b).

So u is selected with degree $d_t(u) = 2$. Observe that no F -edge is incident with u when u is selected. Consequently, either no F -edge is incident with u in the input graph, or an F -edge incident with u is removed from the graph before creation of X . Thus node u saves a coin by Lemma 17 a) resp. by Lemma 17 b).

Since u saves a coin but all nodes of X together save less than two coins by assumption, we get that X saves exactly one coin, i.e. that $\text{pay}_X^\circ = \text{PAY}_X^\circ - 1$ holds. \square

In the proof that a path X which saves only one coin receives at least three coins, we analyze a step for an endpoint of X . In particular, we proceed as follows. Recall that a node matched to create X saves the only coin for X . Hence each other node of X pays a coin, in particular the nodes matched in the *second* step for an endpoint of X , call these nodes x and x' and the according endpoint w . Assuming that w receives only one coin we show that after x and x' become matched all minimum degree nodes are path endpoints, and their degree is one. This implies a contradiction: 1-2-GREEDY never selects a path endpoint. Thus w receives at least two coins. The other endpoint of X receives an additional coin, by Lemma 13.

Lemma 20. *Assume that degrees are bounded by at most $\Delta = 3$. Let X be a path which pays $\text{pay}_X^\circ = \text{PAY}_X^\circ - 1$ coins. Then X receives at least $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ + 1$ coins.*

Proof. We assume that $\text{rcv}_X^\circ < \text{RCV}_X^\circ + 1 = 3$ holds and show a contradiction.

We prepare the argument. Since each endpoint of X receives at least one coin by Lemma 13, the assumption implies that each endpoint of X receives exactly one coin. Moreover, by Lemma 19 the node u selected to create X has degree $d_t(u) = 2$ in the creation step t of X , and u saves the only coin for X . But each M -matched node x of X pays up to $\text{PAY}_x^\circ = \Delta - 2 = 1$ coin, by Lemma 12 b). Consequently, each M -matched node of X but u pays exactly one coin, since $\text{pay}_X^\circ = \text{PAY}_X^\circ - 1$ holds.

To show the desired contradiction, assume that u is matched with node v .

Case 1: Path X is a $\frac{1}{2}$ -path, i.e. we have $m_X = 1$. Let w_u and w_v be the endpoints of X such that we have $\{u, w_u\}, \{v, w_v\} \in M^*$, see Figure 5. Note that we have $w_u \neq w_v$.

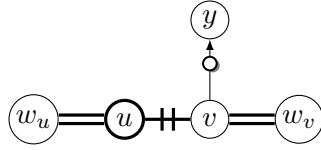


Figure 5: A $\frac{1}{2}$ -path X with a transfer out of v (not all edges are drawn)

Since u and v are the only M -covered nodes of X and u pays no coin, node v pays the only coin, say in transfer (v, y) to path endpoint y . Since in the creation step of X node u is selected with degree $d_t(u) = 2$, in step t the degrees of endpoints w_u, w_v , and y all drop, and no other degrees drop.

Case 1.1: The degree of w_v is at least $d_{t+1}(w_v) \geq 2$ in step $t + 1$. Observe that endpoint w_v still has two incident F -edges in step $t + 1$, since the M^* -edge of w_v is removed in step t . But the M^* -edge of w_v is removed from the graph in step t , hence both F -edges incident with w_v in step $t + 1$ eventually turn out to be transfers. Therefore endpoint w_v receives two coins for X . Since also w_u receives at least one coin by Lemma 13, we obtain $\text{rcv}_X^\circ \geq 3$. A contradiction to our assumption that $\text{rcv}_X^\circ < 3$ holds.

Case 1.2: The degree of w_v is at most $d_{t+1}(w_v) \leq 1$ in step $t + 1$. When edge $\{u, v\}$ is picked the degree of u is exactly $d_t(u) = 2$, i.e. endpoint w_v is not incident with u . Hence the degree of w_v drops by at most one in step t . Consequently, endpoint w_v has degree exactly $d_{t+1}(w) = 1$ after creation of X , which is the new minimum degree in the reduced graph G_{t+1} . But since only the degrees of endpoints w_u, w_v , and y dropped in the creation step t , one of w_u, w_v , and y becomes matched in step $t + 1$. A contradiction is obtained since path endpoints are never matched. The possible configurations are depicted in Figure 6.

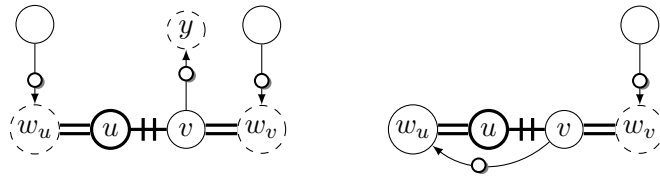


Figure 6: Creating a $\frac{1}{2}$ -path which pays one coin and receives exactly two coins: dashed endpoints have degree one after u and v become matched (not all edges are drawn)

Case 2: Path X has $m_X \geq 2$ edges of M . Since $m_X \geq 2$ holds, after edge $\{u, v\}$ is picked in the creation step of X , there is an M -edge $\{x, x'\}$ of X with the following properties, see Figure 7. One of x and x' , say x , is the M^* -neighbor of an endpoint w of X , and $\{x, x'\}$ is still contained in the graph. In particular, since $\{x, x'\}$ is still contained in the graph whereas $\{u, v\}$ is already removed, we have $u \neq x$ and $u \neq x'$.

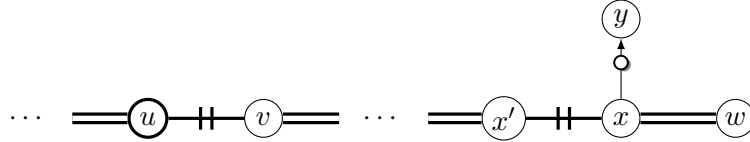


Figure 7: Edge $\{x, x'\}$ is picked in the step for endpoint w of X (not all edges are drawn)

Observe that we may choose w, x , and x' such that x and x' become matched in the second step for an endpoint of X . We denote this step as step s .

Recall that u saves the only coin for X and that x pays a coin, say over transfer (x, y) to endpoint y . Note that both w and y are still connected with x when x becomes matched in step s . Since x is also connected with x' in step s , node x has degree at least $d_s(x) \geq 3 = \Delta$, i.e. we have

$$d_s(x) = 3.$$

To complete the argument, we distinguish cases based on the degree $d_s(w)$ of endpoint w in step s .

Case 2.1: Assume that $d_s(w) = 3 = \Delta$ holds. Observe that two F -edges are incident with w in the reduced graph G_s , call them e and e' . No matter in which step e is removed from the graph (either in step s or later), the degree of w drops to at most one in that step. The same holds for e' . Hence both e and e' are transfers, with one coin moved to w along each one. Thus w receives two coins, contradicting our assumption that each endpoint of X receives exactly one coin.

Case 2.2: Assume that $d_s(w) \leq 2$ holds. Recall that u saves the only coin for X . Since we have $u \neq x'$, node x' pays a coin. Therefore node x' is adjacent to a path endpoint in step s , by Definition 10 (**Transfer**). Since x' is also adjacent to x in step s , node x' has degree at least $d_s(x') \geq 2$. But the degree of x' cannot be larger than the degree $d_s(w)$ of w , since otherwise w would have smaller degree than both x and x' , one of which is selected with minimum degree. We obtain $2 \leq d_s(x') \leq d_s(w) \leq 2$ and hence

$$d_s(x') = d_s(w) = 2.$$

Recall that our goal is to show a contradiction. We prepare the rest of the argument. Since u saves the only coin for X and we have $u \notin \{x, x'\}$, both x and x' pay a coin. In step s , observe the following. By Definition 10 (**Transfer**), each of x and x' is incident with a transfer edge. Since node x' has degree $d_s(x') = 2$ and x' is both incident with a transfer edge and adjacent to the M -neighbor x , we get that the M^* -edge of x' is already removed from the graph. Node x has degree $d_s(x) = 3$, as discussed above, therefore x is adjacent to x' , incident with a transfer edge, and incident with its M^* -edge $\{x, w\}$.

Consequently, in step s all neighbors of x and x' are path endpoints, and only their degrees drop when edge $\{x', x\}$ is picked in step s . The possible configurations are depicted in Figure 8. We distinguish two cases in step s .

- **If x' is not adjacent to w in step s** , then the degree of w drops by one from $d_s(w) = 2$ to $d_{s+1}(w) = 1$ (the three leftmost configurations in Figure 8). Hence either w or another path endpoint is selected in step $s + 1$ and becomes matched. A contradiction, since a path endpoint is never matched.
- Lastly, assume that **x' and w are adjacent** (the rightmost configuration in Figure 8). Then w becomes isolated in step s .

Consider the recipient y of the coin paid by x . Since x' is adjacent to w and x and has degree exactly $d_s(x') = 2$, endpoint y is not adjacent to x' . Therefore the degree of y drops by exactly one in step s . Since the degree of y drops from at least $d_s(y) \geq 2$ to at most $d_{s+1}(y) \leq 1$, by Definition 10, we get that the degree of y is exactly $d_{s+1}(y) = 1$ in step $s + 1$. Furthermore, endpoint y is the only node with degree $d_{s+1}(y) = 1$ in step $s + 1$. Hence y is selected and matched next. A contradiction, since y is a path endpoint. \square

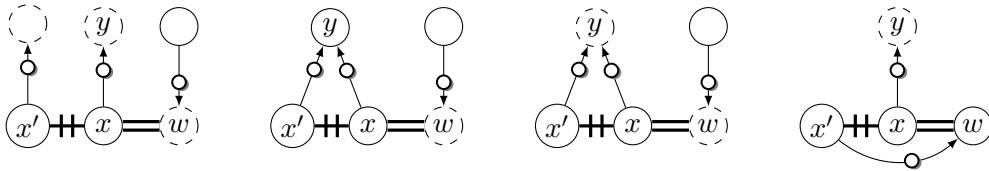


Figure 8: Removing the last M^* -edge at an end of a path: dashed endpoints have degree one after x and x' become matched (not all edges are drawn)

3.2 Maximum Degree $\Delta \geq 4$

This section contains the proof of

Theorem 21. *The 1-2-GREEDY algorithm achieves approximation ratio at least $\frac{\Delta-1}{2\Delta-3}$ for graphs of maximum degree $\Delta \geq 4$.*

Recall that by Lemma 14 it suffices to verify bounds (5 Singleton) and (6 Path), which demand that the balance of a component X is at least

$$\begin{aligned} \text{bal}_X &\geq \text{BAL}_X = -2(\Delta - 2) && \text{if } X \text{ is a singleton and} \\ \text{bal}_X &\geq \text{BAL}_X = 2(\Delta - 1) - 2m_X(\Delta - 2) && \text{if } X \text{ is a path.} \end{aligned}$$

Unlike the case $\Delta = 3$, only moving funds in transfers is insufficient for proving that all local approximation ratios are at least $\frac{\Delta}{2\Delta-3}$. We demonstrate this fact in the following example of an unbalanced $\frac{1}{2}$ -path.

Example 22 (An Unbalanced Path). *Consider the graph in Figure 9, which has maximum degree Δ for some large Δ . In the first $n = \Delta - 4$ steps 1-2-GREEDY creates the $\frac{1}{2}$ -paths X_1, \dots, X_n drawn on the left, each time selecting a node of degree two (drawn in bold). Observe that in the next step, i.e. in step $n + 1$, all degrees in the reduced graph G_{n+1} are at least three: by definition, 1-2-GREEDY picks an arbitrary edge in G_{n+1} . In our example 1-2-GREEDY creates the $\frac{1}{2}$ -path Y drawn on the right. In the last step, the reduced graph is a star centered at node z' , and 1-2-GREEDY creates the singleton Z when selecting node z with degree one.*

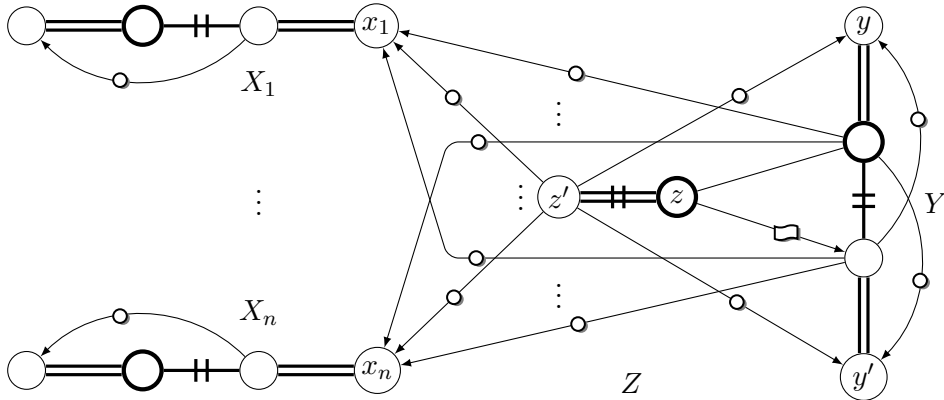


Figure 9: Transfers are insufficient in the proof for general Δ (we have $n = \Delta - 4$ and 1-2-GREEDY selects bold nodes)

The purpose of this example is to demonstrate that the coins moved in transfers are not sufficient for Y to be balanced. First, we argue that the set of transfers is exactly the set of directed edges in the figure.

In step i for $1 \leq i \leq n$ the 1-2-GREEDY algorithm picks the only M -edge in X_i . Observe that the degree of the leftmost path endpoint of $\frac{1}{2}$ -path X_i drops from two to zero. Hence the F -edge incident with this endpoint is a transfer. We call this transfer “internal” since it moves a coin from X_i to X_i . When the $\frac{1}{2}$ -path Y is created, the degree of each endpoint $w \in \{y, y', x_1, \dots, x_n\}$ drops from three to one: all F -edges removed from w in this step are transfers. In the last step, the degree of w drops to zero: all F -edges removed from w in this step are transfers as well.

The $\frac{1}{2}$ -paths X_1, \dots, X_n and singleton Z are $\frac{\Delta-1}{2\Delta-3}$ -balanced. The number of coins owned by each $\frac{1}{2}$ -path X_i is as follows. Path X_i receives two coins from the nodes matched to create Y , one coin from Z , and an “internal” coin from itself. Moreover, path X_i pays an “internal” coin to itself. Hence the number of coins owned by X_i is $\text{rcv}_{X_i}^\circ - \text{pay}_{X_i}^\circ = 4 - 1 \stackrel{\checkmark}{>} 2 = 2(\Delta - 1) - 2(\Delta - 2) \stackrel{m_{X_i}=1}{=} \text{BAL}_{X_i}$, cf. bound (6 Path). Thus X_1, \dots, X_n need no more funds to be balanced. The same is true for singleton Z . Why? Singleton Z pays a coin to each X_i and two coins to Y (and receives zero coins), hence Z owns $\text{rcv}_Z^\circ - \text{pay}_Z^\circ = 0 - (\Delta - 4) - 2 \stackrel{\checkmark}{>} -2(\Delta - 2) = \text{BAL}_Z$ coins, cf. bound (5 Singleton).

However, observe that the $\frac{1}{2}$ -path Y does not own sufficiently many coins. Why? Observe that Y receives two coins from Z as well as two “internal” coins from itself and Y pays two coins to each X_i as well as two “internal” coins to its own endpoints. Therefore we have $\text{rcv}_Y^\circ - \text{pay}_Y^\circ = 4 - 2(\Delta - 3) \stackrel{\checkmark}{<} 2 = 2(\Delta - 1) - 2(\Delta - 2) = \text{BAL}_Y$, cf. (6 Path).

Sketch of the Proof of Theorem 21. In Example 22, the $\frac{1}{2}$ -paths X_i and singleton Z have surpluses, whereas the $\frac{1}{2}$ -path Y has incurred a debt. We show how to redistribute funds to Y and motivate the two main ideas used in the proof of Theorem 21.

Which components should pay funds to Y ? We do not remove funds from the $\frac{1}{2}$ -paths X_i , for two reasons.

First, moving funds from an X_i to Y implies a circular flow of funds, since X_i also receives coins from Y .

Secondly, observe that 1-2-GREEDY “reacts” to the unfavorable creation of a $\frac{1}{2}$ -path, namely Y , by selecting node z of degree one in the next step. Since z has degree one when being selected, no transfer edges leave z and z pays no coins, i.e. the surplus of Z

is considerable. In general, selecting a node of degree one is “optimal” and, intuitively, subtracting funds from nodes matched in an optimal step should be possible *without* their component falling into poverty. To incorporate this “sensitivity” of 1-2-GREEDY into an appropriate definition, we move funds to Y from the nodes matched after creation of Y , namely the nodes matched to create singleton Z .

However, singleton Z does not own enough funds to be able to help Y . Why? Recall that Y owns $\text{rcv}_Y^\circ - \text{pay}_Y^\circ = 4 - 2(\Delta - 4) \stackrel{!}{<} 2 = \text{BAL}_Y$ coins, hence Y needs at least $a = \text{BAL}_Y - (\text{rcv}_Y^\circ - \text{pay}_Y^\circ) = 2(\Delta - 4) - 2$ additional coins in order for $\text{rcv}_Y^\circ - \text{pay}_Y^\circ + a \geq \text{BAL}_Y$ to hold. But Z owns only $\text{rcv}_Z^\circ - \text{pay}_Z^\circ = 0 - (\Delta - 4) - 2$ coins, and supporting Y implies that the balance of Z is no larger than $\text{rcv}_Z^\circ - \text{pay}_Z^\circ - a = -3(\Delta - 4) \stackrel{!}{<} -2(\Delta - 2) = \text{BAL}_Z$: singleton Z cannot support Y without itself being unbalanced. Thus we need to gather funds for Y *also* from the $\frac{1}{2}$ -paths X_i .

We proceed with a strategy of two stages, namely by first *canceling* certain transfers and thereafter inserting a *donation*. In the following two sections we first illustrate each concept by applying it in the above example and then discuss its general properties.

3.2.1 Canceling Transfers

Example. In Example 22, we increase the balance of singleton Z by *canceling* certain transfers leaving Z , i.e. in certain transfers as per Definition 10 we do *not* move coins.

Consider the $\frac{1}{2}$ -path X_1 and recall that X_1 has a surplus. This surplus is caused by endpoint x_1 receiving three coins, namely two coins from path Y and an additional “third” coin from singleton Z . We cancel the third transfer from Z to x_1 . (We show below that, in general, each path endpoint receives at most three coins and the third transfer is uniquely defined.) This reduces the surplus of X_1 , but x_1 still receives two coins from Y and now X_1 owns $\text{rcv}_{X_1}^\circ - \text{pay}_{X_1}^\circ = 3 - 1 = 2 = \text{BAL}_{X_1}$ coins in total. Hence, still X_1 is balanced. We proceed analogously for each endpoint x_i and path X_i . Consequently, singleton Z pays only $\text{pay}_Z^\circ = 2$ coins to the endpoints of Y . Note that no other transfers are canceled, since no other endpoint receives three coins in the first place. (Now each path endpoint in Example 22 receives either one or two coins. These bounds are the exact same bounds as in case $\Delta = 3$: there, each path endpoint receives at least one coin by Lemma 13 and at most two coins since it has at most two incident F -edges in the input graph. We show below that, after canceling transfers, these bounds hold for each path endpoint.)

Towards the Definition. The main aspects of transfer cancellations are the following. We show first that, before cancellation, each endpoint w receives at most three coins (see Lemma 23 a) below) and that the “third” incoming transfer is uniquely defined (see Lemma 23 b) ii.).

Lemma 23. *As a consequence of Definition 10 (Transfer),*

- a) *each path endpoint w receives at most $\text{rcv}_w^{\circ} \leq 3$ coins, and*
- b) *endpoint w receives $\text{rcv}_w^{\circ} = 3$ coins if and only if the the following holds: there are integers $1 \leq t < s$ such that the degree of w drops*
 - i. *from $d_t(w)=3$ to $d_{t+1}(w)=1$ when two incident F -edges are removed and*
 - ii. *from $d_s(w)=1$ to $d_{s+1}(w)=0$ when the last incident F -edge is removed.*

Proof. First we argue that w receives less than three coins if there is a step t_2 when the degree of w is $d_{t_2}(w) = 2$. By Lemma 17 b), no F -edge incident with w removed before step t_2 is a transfer. Moreover, in step t_2 at most two F -edges are still incident with w , i.e. endpoint w receives at most two coins.

Consequently, if endpoint w receives at least three coins, then w never has degree two. This implies the following two facts. First, the degree of w in the input graph G is $d_1(w) \geq 3$, since the degree of w in G is $d_1(w) \geq 2$ by Lemma 8. Secondly, there is a step t_3 when the degree of w is exactly $d_{t_3}(w) = 3$. Why? The degree of w never equals 2, hence we get that there is a step t_3 when the degree of w drops from larger than 2 to below 2. This is only possible if the degree of w drops from exactly $d_{t_3}(w) = 3$, since in each step at most two edges incident with w are removed.

We show a), i.e. that w receives at most three coins. In the step when $d_{t_3}(w)=3$ holds, by Definition 10 (Transfer) endpoint w still receives zero coins. Hence only the remaining three incident edges can be transfer F -edges. Thus w receives at most 3 coins.

We show b). Assume that w receives three coins. In the step when $d_{t_3}(w) = 3$ holds, all remaining edges incident with w must be F -edges, since otherwise w would receive less than three coins. The statement follows (with $t = t_3$ and $s > t_3$), since we have already argued that w never has degree 2. \square

If endpoint w receives the maximum of $\text{rcv}_w^{\circ} = 3$ coins, then by the following definition we cancel the “third” incoming transfer, i.e. the transfer coming from a node getting matched in the step s when the degree of w drops from $d_s(w) = 1$ to $d_{s+1}(w) = 0$, cf. Lemma 23 b) ii.

Definition 24 (Cancel). *Let w be a path endpoint with degree $d_s(w)=1$ and one incident F -edge $\{v, w\}$. If w already receives two coins in step s , then we cancel transfer (v, w) , i.e. we do not move coins over edge $\{v, w\}$.*

Properties. The final set of transfers is given by Definition 10 (Transfer) and Definition 24 (Cancel). In Lemma 25 below we give bounds on the number of coins received by a path endpoint after cancellation of transfers: in particular, we do not cancel transfers in a manner that produces paths without incoming coins, see Lemma 25 b).

Lemma 25. *a) Each path endpoint w receives at most two coins. An endpoint for which an incoming transfer was canceled receives exactly two coins.*

b) Assume that endpoint w belongs to path X . The number rcv_w° and rcv_X° of coins received by w resp. by X are bounded from below by at least

$$\text{RCV}_w^\circ = 1 \text{ resp.}$$

$$\text{RCV}_X^\circ = 2.$$

Proof. a) is a direct consequence of Lemma 23 and Definition 24 (Cancel), since w receives at most three coins, and if so then the third incoming transfer is canceled.

We prove b). Assume that w receives less than two coins. By a), no transfer into w was canceled. In particular, the number of coins received by w does not depend on Definition 24 (Cancel) but only on Definition 10 (Transfer). Therefore we may apply Lemma 13, which shows that w receives at least $\text{rcv}_w^\circ \geq \text{RCV}_w^\circ = 1$ coin.

Like w , the other endpoint of X receives at least one coin. Hence path X receives at least $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ = 2$ coins. \square

Using Lemma 25, we compare coin payments before and after cancellation of transfers. By Lemma 25 a) and b) each path endpoint w receives

$$1 \leq \text{rcv}_w^\circ \leq 2$$

coins. These bounds are the exact same bounds as in case $\Delta = 3$, where we have $1 = \text{RCV}_w^\circ \leq \text{rcv}_w^\circ$ by Lemma 13 resp. $\text{rcv}_w^\circ \leq 2$ since w is incident with at most two F -edges in the input graph.

Moreover, since canceling transfers does not increase the number of coins *payed* by a component X , we also have the following similarities to case $\Delta = 3$. Lemma 12, which gives an upper bound PAY_X° on the coin payment of component X , continues to hold.

Consequently, the lower bounds (3) and (4), see the summary of Section 2.2.2, continue to hold as well: the number of coins owned by component X is at least

$$\text{rcv}_X^\circ - \text{pay}_X^\circ \geq \text{RCV}_X^\circ - \text{PAY}_X^\circ.$$

3.2.2 Donations

Example. In Example 22, recall that after canceling transfers singleton Z pays only $\text{pay}_Z^\circ = 2$ coins. We are now prepared to move further funds to Y . How? The F -edge connecting node z with the M -neighbor of the node selected to create Y is a *donation* edge. In this donation we move an amount of funds to Y which allows *both* Y and Z to be balanced.

For clarity, we refer to funds moved along donations as *bills* instead of coins—however, each coin and bill has equal value κ .

Recall that Y pays $\text{pay}_Y^\circ = 2(\Delta - 4) + 2$ coins to the $\frac{1}{2}$ -paths X_i and to itself. If Z compensates the pay_Y° coins paid by Y by donating

$$\text{rcv}_Y^\square = \text{pay}_Y^\circ$$

many bills to Y , then Y effectively pays nothing. But Y pays no bills itself and receives $\text{rcv}_Y^\circ = 4$ coins, hence Y is balanced with balance $\text{bal}_Y = \text{rcv}_Y^\circ + \text{rcv}_Y^\square - \text{pay}_Y^\circ - \text{pay}_Y^\square = 4 + \text{pay}_Y^\circ - \text{pay}_Y^\circ - 0 = 4 > \text{BAL}_Y$, cf. bound (6 Path). Also, singleton Z pays $\text{pay}_Z^\circ = 2$ coins to the endpoints of Y and donates $\text{pay}_Z^\square = \text{pay}_Y^\circ = 2(\Delta - 3)$ bills, thus Z is balanced with balance $\text{bal}_Z = \text{rcv}_Z^\circ + \text{rcv}_Z^\square - \text{pay}_Z^\circ - \text{pay}_Z^\square = 0 + 0 - 2 - 2(\Delta - 3) = -2(\Delta - 2) = \text{BAL}_Z$.

This stage completes our charging scheme, since all paths X_i are already balanced after canceling transfers.

Towards the Definition. Intuitively, a path X does not depend on a donation if the nodes of X save sufficiently many coins. In particular, path X does not need a donation if the nodes matched in the creation step of X , call them u and v , pay $\text{pay}_{\{u,v\}}^\circ = 0$ coins, as we show in Section 3.2.4. If, however, nodes u and v do pay coins, i.e. if $\text{pay}_{\{u,v\}}^\circ > 0$ holds, then these payments are compensated by a donation in which we move

$$\text{rcv}_{\{u,v\}}^\square \geq \text{pay}_{\{u,v\}}^\circ > 0$$

bills to X , cf. Lemma 27 c) below. Thus, essentially, nodes u and v pay no coins at all and again X turns out to be balanced.

So assume that $\text{pay}_{\{u,v\}}^{\circ} > 0$ holds and that a coin is paid to endpoint w along transfer (x, w) for $x \in \{u, v\}$. In the step after creation of X , say in step $t + 1$, endpoint w has degree at most $d_{t+1}(w) \leq 1$ by Definition 10 (Transfer).¹⁸

Our “trigger” to initiate a donation to X is the existence of an endpoint w with degree *exactly* $d_{t+1}(w) = 1$ after creation of X (we call w a *degree-1 endpoint*), see Figure 10. Why? If a degree-1 endpoint exists after creation of X , then 1-2-GREEDY selects a node u' of degree $d_{t+1}(u') = 1$ next, which is an optimal step by Proposition 1.¹⁹ (We briefly note that $\text{pay}_{\{u,v\}}^{\circ} > 0$ might hold even though there does *not* exist a degree-1 endpoint after creation of X . We characterize these cases in Lemma 33, towards the end of our proof. In such a case we do not depend on the use of a donation: we show that path X is balanced in Lemma 34.)

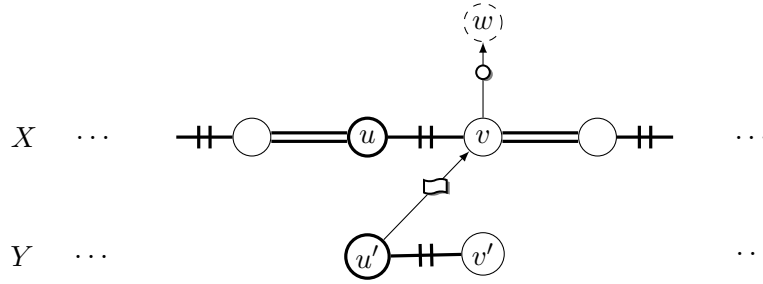


Figure 10: When to initiate a donation (not all edges are drawn, the dashed endpoint w has degree exactly $d_{t+1}(w) = 1$ after u and v become matched)

We initiate a donation from node u' to X only if u' belongs to a component $Y \neq X$ other than X .²⁰ Along which edge does the donation move funds to X ? Since we have $X \neq Y$, an F -edge incident with u' is removed in the creation step t of X when the degree of u' drops from at least $d_t(u') \geq d_t(u) \geq 2$ down to $d_{t+1}(u') = 1$. Moreover, an F -edge incident with node v is removed in step t . Why? If u is selected with degree $d_t(u) = 2$ then no F -edge is incident with u in step t , and if $d_t(u) \geq 3$ holds then

¹⁸E.g. in Example 22 endpoints $y, y', x_1, \dots, x_{\Delta-4}$ have degree one after creation of path Y .

¹⁹E.g. node z in our example has degree one after creation of path Y .

²⁰If u' belongs to X , then we show alter that the optimality of step $t + 1$ helps X in saving funds and being balanced even without an incoming donation.

an F -edge incident with each of u and v must be removed in order for the degree of u' to drop down to $d_{t+1}(u') = 1$. Hence, for the donation we choose edge

$$\{u', v\} \in F.$$

We are now ready to present the definition of donations.

Definition 26 (Donation). *Let X be a path created in step t when 1-2-GREEDY selects node u with degree $d_t(u)$, where $d_t(u) \geq 2$ holds, and matches u with v . Assume that $\text{pay}_{\{u,v\}}^\circ > 0$ holds and that a degree-1 endpoint exists after creation of X . Let nodes u' and v' of component $Y \neq X$ be matched in step $t + 1$, when 1-2-GREEDY selects u' with degree $d_{t+1}(u')=1$.*

- *If $d_t(u) = 2$ holds, then the F -edge $\{u', v\}$ is called a static donation and moves $\Delta - 3$ many bills to v .*
- *If $d_t(u) \geq 3$ holds, then the F -edge $\{u', v\}$ is called a dynamic donation and moves $\text{pay}_{\{u,v\}}^\circ$ many bills to v .*

We denote a donation edge $\{u', v\}$ as (u', v) to indicate its direction and the fact that it pays several bills (instead of only one coin, as for a transfer).

Our intention that (u', v) compensates for coins payed v and u , i.e. that $\text{pay}_{u'}^\square = \text{rcv}_v^\square \geq \text{pay}_{\{u,v\}}^\circ$ holds, is verified as part of the next result, cf. Lemma 27 c). A crucial fact is that the total “debts” of source node u' and its M -neighbor v' cannot become arbitrarily large. In the subsequent section we show that $\text{pay}_{\{u',v'\}}^\circ + \text{pay}_{\{u',v'\}}^\square \leq 2(\Delta - 2)$ holds: intuitively, we generalize the bound $\text{pay}_{\{u',v'\}}^\circ \leq 2(\Delta - 2)$ on coin payments due to Lemma 12 b), cf. Lemma 29 below. We prepare with a discussion of basic properties of donations.

Properties. Observe that an edge cannot be both a transfer and a donation, since a transfer connects an M -matched node with a path endpoint, whereas a donation connects two M -matched nodes. Moreover, donations move funds in the same “direction” as transfers, namely only to paths but never to singletons.

The following properties of a donation are crucial. As we show in Lemma 27 a) below, each node pays coins or bills but not both. Furthermore, in Lemma 27 b) and c) we show that, as claimed above, a donation is sufficient to compensate for the *entire* amount of funds payed by nodes u and v . Also, both u and v pay only coins but no bills.

Lemma 27. *Let x be an M -matched node.*

a) Node x does not pay transfers and donations at the same time, i.e. we have

$$\begin{aligned} \text{pay}_x^\circ > 0 &\Rightarrow \text{pay}_x^\square = 0 \text{ as well as} \\ \text{pay}_x^\square > 0 &\Rightarrow \text{pay}_x^\circ = 0. \end{aligned}$$

Let nodes u and v be matched to create path X , when 1-2-GREEDY selects node u .

- b) Nodes u and v do not pay a donation, i.e. they pay $\text{pay}_{\{u,v\}}^\square = 0$ bills.
 c) If u and v pay $\text{pay}_{\{u,v\}}^\circ > 0$ coins and v receives a donation, then the number of bills moved to v is at least

$$\text{rcv}_v^\square \geq \text{pay}_{\{u,v\}}^\circ.$$

Proof. We prove a). Assume that x pays a donation. It suffices to show that x does not pay transfers. By Definition 26 (Donation), the 1-2-GREEDY algorithm selects node x in a step $t + 1$ when x has degree $d_{t+1}(x) = 1$. In particular, no F -edge is incident with x in step $t + 1$. Hence x pays no transfers by Definition 10 (Transfer).

We prove b). By Definition 26 (Donation), the 1-2-GREEDY algorithm selects the source node of a donation when this node has degree one. However, the nodes u and v have degree at least two in the step when X is created. Therefore neither u nor v pays a donation, i.e. we have $\text{pay}_{\{u,v\}}^\square = 0$.

We prove c). Denote the donation to v by (u', v) . We distinguish cases based on whether (u', v) is a static or a dynamic donation. Assume that X is created in step t .

If u has degree $d_t(u) \geq 3$ in the creation step of X , then by Definition 26 (Donation) the dynamic donation (u', v) moves exactly $\text{pay}_{\{u,v\}}^\circ$ many bills to v , i.e. we have $\text{rcv}_v^\square \geq \text{pay}_{\{u,v\}}^\circ$.

If u has degree $d_t(u) = 2$ in the creation step of X , then by Definition 26 (Donation) the static donation (u', v) moves exactly $\Delta - 3$ bills to v . We have to show that $\text{rcv}_v^\square = \Delta - 3 \stackrel{!}{\geq} \text{pay}_{\{u,v\}}^\circ$ holds.

Since $d_t(u) = 2$ holds, no F -edges are incident with u in step t . Therefore u pays $\text{pay}_u^\circ = 0$ coins by Lemma 17 a) and b). But the F -edge $\{u', v\}$ of donation (u', v) is not a transfer. Consequently, node v saves a coin. In particular, we have $\text{pay}_v^\circ \leq \text{PAY}_v^\circ - 1$, where $\text{PAY}_v^\circ = \Delta - 2$ is an upper bound on the number of coins paid by v , see Lemma 12 b). Therefore nodes u and v pay at most $\text{pay}_{\{u,v\}}^\circ = \text{pay}_u^\circ + \text{pay}_v^\circ \leq \text{PAY}_v^\circ - 1 = \Delta - 3$ coins. \square

3.2.3 Debts of an M -Edge

We continue our discussion of paths. We have already studied in Lemma 27 the payments of the M -edge picked to create a path. Now we turn to analyze the payments of the remaining M -edges. We start by providing some useful notation.

Definition 28. Let x be an M -matched node in M -edge $\{x, x'\}$ in component X . Let $\chi \in \{x, \{x, x'\}, X\}$ be a node, an M -edge, or a component. We denote the number of coins and bills payed by χ as

$$\text{pay}_\chi = \text{pay}_\chi^\circ + \text{pay}_\chi^\square.$$

By PAY_χ we denote an upper bound on pay_χ° , i.e. we have $\text{pay}_\chi \leq \text{PAY}_\chi$.

Lemma 29. Consider an M -edge $\{x, x'\}$. The number $\text{pay}_{\{x, x'\}}$ of coins and bills payed by $\{x, x'\}$ is bounded by at most

$$\text{PAY}_{\{x, x'\}} = 2(\Delta - 2).$$

Before the proof we emphasize that Lemma 29 applies to *each* M -edge, in particular to a singleton: an immediate consequence is that balance bound (5 Singleton) holds, as we show next. Thus after proving Lemma 29 it remains to verify balance bound (6 Path).

Corollary 30. If Lemma 29 holds, then a singleton X pays at most $\text{pay}_X \leq 2(\Delta - 2)$ coins and bills. In particular, the balance bound (5 Singleton) holds.

Proof. Since a singleton X receives $\text{rcv}_X = 0$ coins and bills (recall that transfers as well as donations move funds only to paths), we have $\text{bal}_X = \text{rcv}_X - \text{pay}_X \geq -\text{PAY}_X \stackrel{\text{Lemma 29}}{=} -2(\Delta - 2) \stackrel{(5 \text{ Singleton})}{=} \text{BAL}_X$. \square

Now consider an M -edge $\{x, x'\}$ of a path. Lemma 29 generalizes Lemma 12 b), which states that edge $\{x, x'\}$ pays at most $\text{pay}_{\{x, x'\}}^\circ \leq \text{PAY}_{\{x, x'\}}^\circ = 2(\Delta - 2)$ coins, in the following two ways. First, Lemma 29 additionally includes bills in the same bound. Secondly, by Lemma 12 b) each of nodes x and x' pays at most $\Delta - 2$ coins, whereas Lemma 29 does *not* imply that the maximum share of funds payed by x resp. x' is equal, i.e. one of both nodes might pay all funds while the other pays nothing.

Proof of Lemma 29. Throughout the rest of this section we prove Lemma 29. We distinguish two cases based on whether a node in edge $\{x, x'\}$ pays a donation. If no donation must be payed, then by means of previous results the argument reduces to the

analysis of a singleton. Otherwise we have to investigate coin payments of both M -edges involved in a donation.

Case 1: Paying no Donation. First, assume that nodes x and x' do not pay a donation, i.e. that $\text{pay}_{\{x,x'\}}^{\square} = 0$ holds. We have to show that

$$\text{pay}_{\{x,x'\}} = \text{pay}_{\{x,x'\}}^{\circ} \leq 2(\Delta - 2)$$

holds. If edge $\{x, x'\}$ belongs to a path, then by Lemma 12 b) we have $\text{pay}_{\{x,x'\}}^{\circ} \leq \text{PAY}_{\{x,x'\}}^{\circ} = 2(\Delta - 2)$ and are done.

So assume that $\{x, x'\}$ is the M -edge of a singleton, call it X . Recall that X pays at most $\text{pay}_{\{x,x'\}}^{\circ} \leq \text{PAY}_{\{x,x'\}}^{\circ} = 2(\Delta - 1)$ coins by Lemma 12 a). It suffices to show that X saves two coins, since then $\text{pay}_{\{x,x'\}}^{\circ} \leq \text{PAY}_{\{x,x'\}}^{\circ} - 2 = 2(\Delta - 2)$ holds, as desired.

Assume that singleton X saves at most one coin. We show a contradiction, which we prepare with the following two facts. First, by assumption, nodes x and x' pay at least $\text{pay}_{\{x,x'\}}^{\circ} \geq \text{PAY}_{\{x,x'\}}^{\circ} - 1 = 2(\Delta - 1) - 1$ coins. In particular, we have $\text{pay}_{\{x,x'\}}^{\circ} \geq 1$, i.e. a transfer leaves one of x and x' . Secondly, when 1-2-GREEDY creates singleton X , say in step t , both x and x' have degree at least $\Delta - 1$. Why? Since otherwise one of x and x' has degree at most $\Delta - 2$ and by Lemma 17 a) and b) saves two coins, which would contradict our assumption.

In the creation step t of X , consider an endpoint w receiving a transfer from x or x' . Since both x and x' have degree at least $\Delta - 1 \geq 3$, endpoint w has degree at least $d_t(w) \geq 3$ as well. Moreover, endpoint w has degree at least $d_{t+1}(w) \geq 1$ after creation of X , since at most two edges incident with w are removed during step t . But w also has degree at most $d_{t+1}(w) \leq 1$ in step $t + 1$, since it receives a transfer from x or x' . (Since the degree of w drops by two, an F -edge incident with each of x and x' is removed. Therefore w even receives a transfer from each of x and x' , see Figure 11.) So $d_{t+1}(w) = 1$ holds. Since w never becomes matched, another node y of degree $d_{t+1}(y) = 1$ is selected in step $t + 1$. Since y had degree at least $d_t(y) \geq 3$ in the creation step of X as well, two edges $\{x, y\}$ and $\{x', y\}$ connecting x and x' with y are removed in the creation step of X . Both $\{x, y\}$ and $\{x', y\}$ are not transfers, since nodes x, x' , and y are M -matched. Consequently, each of x and x' saves a coin by Lemma 17 c), a contradiction.

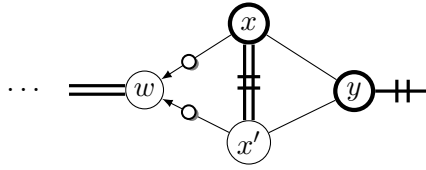


Figure 11: A singleton saves two coins (not all edges are drawn, the “third” edge incident with w does not have to be an M^* -edge)

Case 2: Paying a Donation. Here, we do not distinguish if the component of edge $\{x, x'\}$ is a path or a singleton and conduct a unified analysis. For convenience, we use the naming conventions introduced in Section 3.2.2, i.e. we let

$$\{x, x'\} = \{u', v'\}$$

and assume that node u' in component Y pays a donation (u', v) to node v in path X , cf. Figure 12. In order to prove Lemma 29, we have to show that nodes u' and v' pay at most

$$\text{pay}_{\{u', v'\}} \leq 2(\Delta - 2) \tag{11}$$

coins and bills.

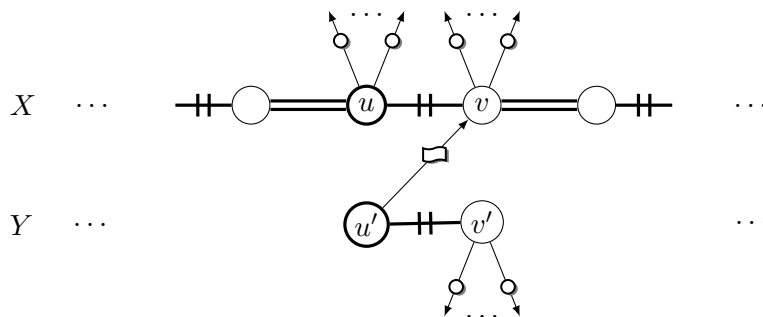


Figure 12: Debts of an M -edge $\{u', v'\}$ (not all edges are drawn)

We reformulate (11). Observe that node u' pays only bills while neighbor v' pays only coins, i.e. that

$$\text{pay}_{\{u',v'\}} = \text{pay}_{u'}^{\square} + \text{pay}_{v'}^{\circ}$$

holds. Why? First, node u' donates $\text{pay}_{u'}^{\square} > 0$ bills along (u', v) , hence node u' pays $\text{pay}_{u'}^{\circ} = 0$ coins by Lemma 27 a). Secondly, node v' does not pay a donation by Definition 26 (Donation), hence we have $\text{pay}_{v'}^{\square} = 0$.

Moreover, by Lemma 27 c) the $\text{pay}_{u'}^{\square} = \text{rcv}_{v'}^{\square}$ donated bills compensate for the $\text{pay}_{\{u,v\}}^{\circ}$ coins paid by nodes u and v , i.e. we have $\text{pay}_{\{u,v\}}^{\circ} \leq \text{pay}_{u'}^{\square}$. Consequently we have

$$\text{pay}_{\{u,v\}}^{\circ} + \text{pay}_{v'}^{\circ} \leq \underbrace{\text{pay}_{u'}^{\square} + \text{pay}_{v'}^{\circ}}_{=\text{pay}_{\{u',v'\}}} \stackrel{(11)}{\leq} 2(\Delta - 2),$$

where (11) is to be shown. The proof of (11) proceeds in two phases.

Phase 1: We ignore the middle term $\text{pay}_{\{u',v'\}}$ and show $\text{pay}_{\{u,v\}}^{\circ} + \text{pay}_{v'}^{\circ} \leq 2(\Delta - 2)$.

This is a direct consequence of Lemma 31 below, which states a more general fact: Lemma 31 does not require that v receives a donation from u' or even that u' belongs to another component than v . Instead, we merely assume that a degree-1 endpoint exists after creation of the path X of u and v , which is a “trigger” to initiate a donation, cf. Definition 26.

The proof of Lemma 31 uses Lemma 31₂ and Lemma 31₃ analyzing cases $d_t(u)=2$ resp. $d_t(u) \geq 3$.

Phase 2: Assume that the inequality of the first phase holds. We show that replacing $\text{pay}_{\{u,v\}}^{\circ}$ with $\text{pay}_{u'}^{\square}$ does not violate the $2(\Delta - 2)$ -bound.

We carry out the second phase now.

Dynamic Donation. Recall that u' donates $\text{pay}_{u'}^{\square}$ bills to v . If (u', v) is a dynamic donation, then by Definition 26 we have $\text{pay}_{u'}^{\square} = \text{pay}_{\{u,v\}}^{\circ}$ and (11) follows.

Static Donation. So assume that (u', v) is a static donation. By Definition 26, the 1-2-GREEDY algorithm selects node u with degree $d_t(u) = 2$. We apply Lemma 31₂ b) and obtain that $\text{pay}_{\{u,v\}}^{\circ} + \text{pay}_{v'}^{\circ} \leq (\Delta - 3) + \text{pay}_{v'}^{\circ} \leq 2(\Delta - 2)$ holds. Since we move $\Delta - 3 = \text{pay}_{u'}^{\square}$ coins along (u', v) by Definition 26, we may replace $(\Delta - 3)$ with $\text{pay}_{u'}^{\square}$ and (11) follows.

Lemma 31. *Assume that in step t the 1-2-GREEDY algorithm creates path X when selecting node u and picking edge $\{u, v\}$. Furthermore, assume that a degree-1 endpoint exists in step $t + 1$, when 1-2-GREEDY selects node u' and picks edge $\{u', v'\}$. We have*

$$\text{pay}_{\{u,v\}}^{\circ} + \text{pay}_{v'}^{\circ} \leq 2(\Delta - 2).$$

In the proof we distinguish if 1-2-GREEDY selects node u with degree $d_t(u) = 2$ or with degree $d_t(u) \geq 3$, see Lemma 31₂ resp. Lemma 31₃.

Lemma 31₂. *Consider Lemma 31 and assume that 1-2-GREEDY picks edge $\{u, v\}$ when u has degree $d_t(u) = 2$. If edge $\{u', v'\}$ belongs to*

a) *path X then we have*

$$\begin{aligned} \text{pay}_{\{u,v\}}^{\circ} &\leq \Delta - 2 && \text{and} \\ \text{pay}_{v'}^{\circ} &\leq \Delta - 2. \end{aligned}$$

b) *component $Y \neq X$ then we have*

$$\begin{aligned} \text{pay}_{\{u,v\}}^{\circ} &\leq \Delta - 3 && \text{and} \\ \text{pay}_{v'}^{\circ} &\leq \Delta - 1. \end{aligned}$$

Proof. To prepare the proof, consider the creation step t of path X . Observe that u is not incident with an F -edge, since u is incident with an M -edge and an M^* -edge and has degree $d_t(u) = 2$. Therefore node u pays $\text{pay}_u^{\circ} = 0$ coins by Lemma 17 a) and b). Hence $\text{pay}_{\{u,v\}}^{\circ} = \text{pay}_v^{\circ}$ holds and it suffices to bound pay_v° .

We prove a). Both v and v' belong to path X , hence the number of coins payed by node $v \in \{v, v'\}$ is at most $\text{pay}_v^{\circ} \leq \text{PAY}_v^{\circ} = \Delta - 2$ by Lemma 12 b). We obtain $\text{pay}_{\{u,v\}}^{\circ} = \text{pay}_v^{\circ} \leq \Delta - 2$ as well as $\text{pay}_{v'}^{\circ} \leq \Delta - 2$.

We prove b). Edge (u', v) is not a transfer, since both u' and v are M -matched nodes. Consequently, node v saves a coin by Lemma 17 c). Hence we obtain that v pays at most $\text{pay}_v^{\circ} \leq \text{PAY}_v^{\circ} - 1 \leq \Delta - 3$ coins, cf. Lemma 12 b). Therefore we have $\text{pay}_{\{u,v\}}^{\circ} = \text{pay}_v^{\circ} \leq \Delta - 3$.

Now consider node v' and recall that the M -edge incident with v' is not a transfer as well. By the degree constraint we get $\text{pay}_{v'}^{\circ} \leq \Delta - 1$. \square

Observe that the proof of Lemma 31₂ could be established by analyzing each of nodes u, v , and v' individually. However, in case 1-2-GREEDY selects node u with

degree $d_t(u) \geq 3$, the analysis has to be more complex. Why? Since 1-2-GREEDY picks an arbitrary edge, we have less control over which transfers are incident with the three nodes. In particular, bounds on $\text{pay}_{\{u,v\}}^\circ$ and on pay_v° have to be chosen “dynamically” depending on each other. We control dependencies by investigating endpoints neighboring the three nodes (in the input graph). For convenience, we categorize these endpoints using the following notation.

Definition 32 (Types of Endpoints). *Consider the creation step t of path X , when 1-2-GREEDY selects node u with degree $d_t(u) \geq 3$ and matches u with v . We denote by (see Figure 13 for examples)*

- \mathcal{W} the set of path endpoints w with degree at least $d_t(w) \geq d_t(u) \geq 3$ in step t ,
- $W \subseteq \mathcal{W}$ the set of \mathcal{W} -endpoints neighboring u or v in step t ,
- $W_\delta = \{w \in W : d_{t+1}(w) = \delta\}$ the set of W -endpoints with degree δ in step $t + 1$,
- $W_{\geq 3} = W \setminus (W_1 \cup W_2)$ the set of W -endpoints with degree at least 3 in step $t + 1$,
- $W_1^f = \{w \in W_1 : |\{\{w, u\}, \{w, v\}\} \cap F| = f\}$ the set of degree-1 endpoints in step $t + 1$ being connected (in G) with u or v by $f \in \{1, 2\}$ edges in F , and
- $\mathcal{E}(W) = \{\{x, y\} \in E : x \in \{u, v\}, y \in W\}$ the set of all edges connecting nodes u and v with all W -endpoints.

Note that $W_{\geq 3}$, W_2 , and W_1 form a partition of W and that W_1^2 and W_1^1 form a partition of W_1 , i.e. sets $\mathcal{W} \setminus W, W_{\geq 3}, W_2, W_1^2$, and W_1^1 are pairwise disjoint.

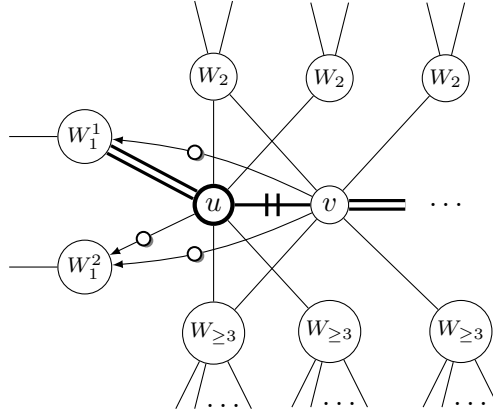


Figure 13: Examples for types of endpoints (an endpoint is labeled with the set it belongs to, endpoints in $\mathcal{W} \setminus W$ are not drawn)

Lemma 31₃. Consider Lemma 31 and assume that 1-2-GREEDY picks edge $\{u, v\}$ when u has degree $d_t(u) \geq 3$. We have

$$\text{pay}_{\{u,v\}}^\circ = 2|W_1^2| + |W_1^1| \quad \text{and} \quad (\text{a})$$

$$\text{pay}_{v'}^\circ \leq |W_1^1| + |W_2| \quad (\text{b})$$

as well as

$$2|W_1^2| + 2|W_1^1| + |W_2| \leq |\mathcal{E}(W)| \quad (\text{c})$$

$$\leq 2(\Delta - 2). \quad (\text{d})$$

Proof. We prove (a), where we show that nodes u and v pay $\text{pay}_{\{u,v\}}^\circ = 2|W_1^2| + |W_1^1|$ coins. To count the number of coins paid by u and v , we first study which endpoints do not receive a transfer from u or v .

- ✗ An endpoint $w \in (\mathcal{W} \setminus W) \cup W_{\geq 3}$ has degree $d_{t+1}(w) \geq 3$ in step $t + 1$ and hence does not receive a transfer from u or v . Why? For $w \in W_{\geq 3}$ this follows directly from Definition 32, for $w \in \mathcal{W} \setminus W$ observe that w is not adjacent to u or v and consequently the degree of w does not drop in step t .
- ✗ Also, an endpoint $w \in W_2$ receives no transfer from u or v , since its degree is $d_{t+1}(w) = 2$ in step $t + 1$.
- ✓ Hence only an endpoint $w \in W_1$ might receive a transfer from u or v , since its degree is exactly $d_{t+1}(w) = 1$ in step $t + 1$. In the creation step t of X , either two F -edges incident with w are removed, i.e. we have $w \in W_1^2$, or one incident F -edge and an incident M^* -edge is removed, i.e. we have $w \in W_1^1$. Since all these removed F -edges are transfers, we get that nodes u and v pay exactly $\text{pay}_{\{u,v\}}^\circ = 2|W_1^2| + |W_1^1|$ coins.

To show (b), we have to prove that node v' pays at most $\text{pay}_{v'}^\circ \leq |W_1^1| + |W_2|$ coins. We proceed similar to (a), i.e. we study which endpoints do not receive a transfer from v' .

- ✗ An endpoint $w \in (\mathcal{W} \setminus W) \cup W_{\geq 3}$ does not receive a transfer from v' . Why? Endpoint w has degree at least $d_{t+1}(w) \geq 3$ in step $t + 1$. Hence when v' becomes matched, namely when 1-2-GREEDY selects node u' with degree $d_{t+1}(u') = 1$, the degree of w drops by at most one and thus to at least $d_{t+2}(w) \geq 2$ (but degree at most one would be required in order to receive a transfer from v').

- ✗ Next, we study endpoints in W_1^2 . Here, Definition 24 (Cancel) comes into play. An endpoint $w \in W_1^2$ does not receive a transfer from v' , since w already receives two transfers from u and v , namely over the $f = 2$ many F -edges removed in step t : any additional transfer (v', w) is canceled.
- ✓ Hence v' might pay a transfer only to endpoints in W_1^1 and W_2 , and (b) follows. (We note that, in particular, an endpoint $w \in W_1^1 \cup W_2$ receives *less* than two credits from u or v , i.e. further credits are *not* canceled, and we have $d_{t+1}(w) \leq 2$, i.e. the degree of w might drop to at most one when edge $\{v', w\}$ is removed.)

We prove (c), i.e. that $2(|W_1^1| + |W_1^2|) + |W_2| \leq |\mathcal{E}(W)|$ holds.²¹ For each endpoint $w \in W_1^1 \cup W_1^2$, recall that two edges connecting w with u and v are removed when in the creation step t of path X the degree of w drops from at least $d_t(w) \geq d_t(u) \geq 3$ to $d_{t+1}(w) = 1$. These two edges belong to $\mathcal{E}(W)$. Moreover, each endpoint $w \in W_2$ is connected with u or v by at least one edge in $\mathcal{E}(W)$, since otherwise the degree of w would not drop to 2 during step t .

We prove (d), i.e. that $|\mathcal{E}(W)|$ is bound from above by $2(\Delta - 2)$. To see this, observe first that in the creation step of X two edges incident with the selected node u do not connect u with an endpoint: edge $\{u, v\}$ is an M -edge and edge $\{u, u'\}$ is removed when in the creation step t of X the degree of u' drops from at least $d_t(u') \geq d_t(u) \geq 3$ to $d_{t+1}(u') = 1$. Consequently, at most $\Delta - 2$ edges connect u with an endpoint. Analogously, edges $\{v, u\}$ and $\{v, u'\}$ do not connect node v with an endpoint, i.e. at most $\Delta - 2$ edges connect v with an endpoint. The statement follows. \square

3.2.4 Amortization for Paths

Throughout this section we verify balance bound (6 Path), i.e. that the balance of a path X is at least $\text{bal}_X \geq \text{BAL}_X = 2(\Delta - 1) - m_X \cdot 2(\Delta - 2)$.

As a consequence of Lemma 29, path X pays at most

$$\text{PAY}_X = m_X \cdot 2(\Delta - 2)$$

coins and bills, since X has exactly m_X edges in M . Also, the path endpoints of X receive at least $\text{RCV}_X^\circ = 2$ coins by Lemma 13. Thus X owns at least $\text{rcv}_X - \text{pay}_X \geq$

²¹Note that $\text{pay}_{\{u,v\}}^\circ + \text{pay}_{v'}^\circ \leq |\mathcal{E}(W)|$ does not trivially hold, since without further investigation we can only make use of bounds $\text{pay}_{\{u,v\}}^\circ \leq 2(\Delta - 2)$ and $\text{pay}_{v'}^\circ \leq \Delta - 1$, whereas we have $|\mathcal{E}(W)| \leq 2(\Delta - 1)$.

$\text{rcv}_X^\circ - \text{pay}_X \geq \text{RCV}_X^\circ - \text{PAY}_X = 2 - m_X \cdot 2(\Delta - 2)$ coins and bills. To verify bound (6 Path) we have to show that the balance of X is at least

$$\begin{aligned} \text{BAL}_X &= 2(\Delta - 1) - m_X \cdot 2(\Delta - 2) \\ &= \text{RCV}_X^\circ - \text{PAY}_X + 2(\Delta - 2). \end{aligned}$$

I.e. we have to show that in our charging scheme the balance of X is increased by at least

$$2(\Delta - 2)$$

above $\text{RCV}_X^\circ - \text{PAY}_X$.

The proof is organized in three parts as follows. Assume that in the creation step of X the 1-2-GREEDY algorithm selects node u and matches u with node v . We distinguish if u and v pay no coins (part 1), if they do pay coins and v receives a donation (part 2), or if they pay coins but do not receive a donation (part 3).

Part 1: Saving Coins. We show that—as we have claimed in Section 3.2.2—path X is balanced if nodes u and v pay $\text{pay}_{\{u,v\}}^\circ = 0$ coins. Recall that by Lemma 27 b) nodes u and v pay $\text{pay}_{\{u,v\}}^\square = 0$ bills. We get $\text{pay}_{\{u,v\}} = 0 = \text{PAY}_{\{u,v\}} - 2(\Delta - 2)$, since both nodes pay at most $\text{PAY}_{\{u,v\}} = 2(\Delta - 2)$ coins and bills in the first place, by Lemma 29. Hence the “debts” of path X are at most $\text{pay}_X = \sum_{\{x,x'\}} \text{pay}_{\{x,x'\}} \leq \text{PAY}_X - 2(\Delta - 2)$, where we sum over all M -edges $\{x, x'\}$ of X . Consequently, the balance of X is at least

$$\begin{aligned} \text{bal}_X &= \text{rcv}_X - \text{pay}_X \\ &\geq \text{rcv}_X^\circ - \text{pay}_X \\ &\geq \text{RCV}_X^\circ - \text{PAY}_X + 2(\Delta - 2) \\ &= \text{BAL}_X \end{aligned}$$

and we are done.

Part 2: Receiving Bills. Assume that nodes u and v pay $\text{pay}_{\{u,v\}}^\circ > 0$ coins, and that v receives a donation (u', v) . By Lemma 27 b) nodes u and v pay $\text{pay}_{\{u,v\}}^\square = 0$ bills, therefore we have $\text{pay}_{\{u,v\}} = \text{pay}_{\{u,v\}}^\circ$. By Lemma 27 c) in the donation we

move $\text{rcv}_v^\square \geq \text{pay}_{\{u,v\}}^\circ$ bills to X . Therefore the balance of X is at least (the sum is over M -edges $\{x, x'\}$ in X)

$$\begin{aligned}
 \text{bal}_X &= \text{rcv}_X - \text{pay}_X \\
 &= \left(\underbrace{\text{rcv}_X^\circ}_{\geq \text{RCV}_X^\circ} + \underbrace{\text{rcv}_X^\square}_{= \text{rcv}_v^\square} \right) - \left(\underbrace{\text{pay}_{\{u,v\}}^\circ}_{= \text{pay}_{\{u,v\}}^\circ} + \underbrace{\sum_{\{x,x'\} \neq \{u,v\}} \text{pay}_{\{x,x'\}}^\circ}_{\leq (m_X-1) \cdot \text{PAY}_{\{x,x'\}}^\circ} \right) \\
 &\geq (\text{RCV}_X^\circ + \text{rcv}_v^\square) - (\text{pay}_{\{u,v\}}^\circ + (\text{PAY}_X - 2(\Delta - 2))) \\
 &= \text{RCV}_X^\circ - \text{PAY}_X + 2(\Delta - 2) + (\text{rcv}_v^\square - \text{pay}_{\{u,v\}}^\circ) \\
 &\geq \text{BAL}_X.
 \end{aligned}$$

Part 3: No Compensation. Assume that nodes u and v pay $\text{pay}_{\{u,v\}}^\circ > 0$ coins, but that v *does not* receive a donation. Then, in particular, there does not exist a degree-1 endpoint after creation of path X (recall that by Definition 26 the existence of a degree-1 endpoint after creation is our “trigger” to initiate a donation). In the next result we identify the exact “configuration” of endpoints neighboring u and v and determine $\text{pay}_{\{u,v\}}^\circ$, see i. to viii. of Lemma 33 b).

Lemma 33. *Consider the creation step t of path X , where nodes u and v become matched and 1-2-GREEDY selects u with degree $d_t(u)$. Assume that u and v pay $\text{pay}_{\{u,v\}}^\circ > 0$ coins and let (x, w) for $x \in \{u, v\}$ be a transfer.*

- a) *If $d_t(u) \geq 3$ holds, then a degree-1 endpoint exists after creation of X .*
- b) *If $d_t(u) = 2$ holds, then a degree-1 endpoint exists after creation of X , unless the following holds (see Figure 14 for an illustration of this exception):*
 - i. *after creation of X endpoint w has degree $d_{t+1}(w) = 0$,*
 - ii. *in the creation step of X endpoint w has degree $d_t(w) = 2$,*
 - iii. *no transfer leaves u , i.e. we have $\text{pay}_u^\circ = 0$,*
 - iv. *transfer (x, w) leaves v , i.e. we have $(x, w) = (v, w)$,*
 - v. *we have $\{u, w\} \in M^*$,*
 - vi. *in the creation step of X all other endpoints w_1, \dots, w_k neighboring v (i.e. we have $w \notin \{w_1, \dots, w_k\}$) have degree at least $d_t(w_i) \geq 3$,*
 - vii. *node v pays no other transfer besides (v, w) , i.e. we have $\text{pay}_v^\circ = 1$, and*
 - viii. *we have $\text{pay}_{\{u,v\}}^\circ = 1$.*

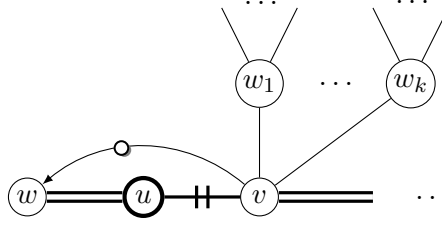


Figure 14: No degree-1 endpoint exists after creation of path X (endpoint w could be one of the w_i)

Proof. By Definition 10 (Transfer), endpoint w has degree at most $d_{t+1}(w) \leq 1$ in the step after creation of X . Observe that in the creation step of X endpoint w is not isolated, since (x, w) is an incoming transfer and hence edge (x, w) is still incident with w .

a) If $d_t(u) \geq 3$ holds, then the degree of w drops from at least $d_t(w) \geq 3$ to at least $d_{t+1}(w) \geq 1$, since at most two edges incident with w are removed. Since we also have $d_{t+1}(w) \leq 1$ by definition of transfer (x, w) , endpoint w has degree exactly $d_{t+1}(w)=1$ after creation of X , i.e. endpoint w is a degree-1 endpoint.

b) Now assume that $d_t(u) = 2$ holds. Since we have $d_{t+1}(w) \leq 1$ after creation of X , to prove the statement it suffices to study the case that no degree-1 endpoint exists after creation. In particular, endpoint w has degree $d_{t+1}(w) = 0$ after creation. Observe that i. holds. It remains to verify ii. to viii.

- ii. Endpoint w has degree $d_t(w) = 2$ in the creation step, since the degree is $d_t(w) \geq 2$ by Lemma 8 a) and if the degree was at least $d_t(w) \geq 3$ then it could not drop to $d_{t+1}(w) = 0$ after creation, as is fact by i.
- iii. Since u has degree $d_t(u) = 2$ in the creation step of X , node u is not incident with an F -edge, as would be required in order to pay a transfer.
- iv. Since transfer (x, w) does not leave u by iii., transfer (x, w) leaves v .
- v. Since during the creation step of X the degree of w drops by two (from $d_t(w) \geq d_t(u) = 2$ to $d_{t+1}(w) = 0$), an edge connecting w with each of u and v is removed. One of both edges must be the M^* -edge incident with w . Since transfer $(x, w) = (v, w)$ is an F -edge, we obtain $\{u, w\} \in M^*$.
- vi. Observe that for each endpoint w_i we have $d_{t+1}(w_i) \geq 1$ after creation of X , since in the creation step of X we have $d(w_i)_t \geq d_t(u) = 2$ and w_i is not connected with u . Since we assumed that after creation of X no degree-1 endpoint exists

and w_i is not isolated, we obtain $d_{t+1}(w_i) \geq 2$. But an edge incident with w_i is removed in the creation step of X . Hence we get $d_t(w_i) \geq 3$.

vii. This follows from vi. Why? If each w_i has degree at least three $d_t(w_i) \geq 3$ creation of X , then no w_i receives a transfer from v , since the degree of each w_i drops by exactly one (recall that u is adjacent only to v and w). Consequently, endpoint w receives the only transfer from v .

viii. By iii. and vii. we have $\text{pay}_{\{u,v\}}^\circ = \text{pay}_u^\circ + \text{pay}_v^\circ = 1$. □

Recall that by Lemma 27 b) nodes u and v pay no bills. Hence we have $\text{pay}_{\{u,v\}} = \text{pay}_{\{u,v\}}^\circ + \text{pay}_{\{u,v\}}^\square = 1$. Moreover, by Lemma 29 the total payments of u and v are bounded by at most $\text{PAY}_{\{u,v\}} = 2(\Delta - 2)$, thus we have $\text{pay}_{\{u,v\}} = \text{PAY}_{\{u,v\}} - 2(\Delta - 2) + 1$. Consequently, paths X pays at most

$$\text{pay}_X = \text{pay}_{\{u,v\}} + \sum_{\{x,x'\} \neq \{u,v\}} \text{pay}_{\{x,x'\}} \leq \text{PAY}_X - 2(\Delta - 2) + 1$$

coins and bills and the funds owned by X are at least

$$\begin{aligned} \text{rcv}_X - \text{pay}_X &\geq \text{rcv}_X^\circ - \text{pay}_X \\ &\geq \text{RCV}_X^\circ - \text{PAY}_X + 2(\Delta - 2) - 1 \\ &= \text{BAL}_X - 1. \end{aligned}$$

I.e. we have failed the required balance for X by exactly one.

How do we show that the balance of X is actually increased by at least one? We show that X receives at least $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ + 1$ coins, i.e. at least one coin more than assumed above. Or we show that there is an M -edge $\{x,x'\} \neq \{u,v\}$ of X with $\text{pay}_{\{x,x'\}} \leq \text{PAY}_{\{x,x'\}} - 1$, which reduces the payments of X by at least one and thus to at most $\text{pay}_X \leq \text{PAY}_X - 2(\Delta - 2)$. Hence the following result completes the proof of Theorem 21.

Lemma 34. *Assume that 1-2-GREEDY creates path X in step t when selecting node u and picking edge $\{u,v\}$. If no degree-1 endpoint exists in step $t + 1$, then*

- a) *path X receives at least $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ + 1$ coins or*
- b) *there is an M -edge $\{x,x'\} \neq \{u,v\}$ of X which pays at most $\text{pay}_{\{x,x'\}} \leq \text{PAY}_{\{x,x'\}} - 1$ coins and bills.*

Proof. To prepare the proof, observe that to show that $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ + 1$ holds it suffices to identify an endpoint of X which receives at least two coins, since by Lemma 13 the

other endpoint of X receives an additional coin. We distinguish cases by the number m_X of M -edges of X .

Case 1: $m_X = 1$. We show that an endpoint of X receives at least two coins. First, we consider the creation step t of X and argue that in the subsequent step $t + 1$ there is an endpoint of X which is not isolated. Since after creation of X there does not exist a degree-1 endpoint, by Lemma 33 b) the 1-2-GREEDY algorithm selects node u with degree $d_t(u) = 2$ to create X , see Figure 15. Observe that the M^* -neighbor of v , call it w , has degree at least $d_t(w) \geq d_t(u) = 2$ as well. Since $m_X = 1$ holds, node w must be an endpoint of X , and since we have $d_t(u) = 2$ endpoint w is not adjacent to u in step t . Thus during step t the degree of w drops by at most one and to at least $d_{t+1}(w) \geq 1$.

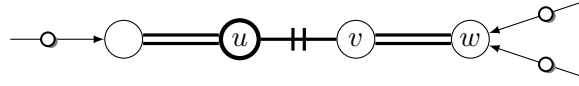


Figure 15: Case 1: An endpoint w of X receives two coins (not all edges are drawn)

Moreover, in step $t + 1$ all non-isolated path endpoints have degree at least two, since there does not exist a degree-1 endpoint after creation. This holds in particular for w . But since after creation of X endpoint w is incident with at least two F -edges, endpoint w eventually receives at least two coins, namely over those two F -edges of w which are removed last.

Case 2: $m_X \geq 2$. Recall that no degree-1 endpoint exists after creation of X , hence by Lemma 33 b) (in particular by v.) an M^* -edge $\{u, w\}$ connects u with an endpoint w of X . Consequently, the creation step of X is also the first step for an endpoint of X , namely endpoint w .

Consider the second step for an endpoint of X , i.e. the step s when for the second time an M^* -edge incident with an endpoint of X is removed from the (reduced) graph. Denote by x the selected node and by x' the neighbor matched with x . Since $m_X \geq 2$ holds we have $\{u, v\} \neq \{x, x'\}$, cf. Figure 16. We distinguish the cases that x does not pay a donation, or x pays a static or a dynamic donation.

Case 2.1: No Donation. Since x pays no donation, both nodes pay $\text{pay}_{\{x, x'\}}^{\square} = 0$ bills. Thus they pay only coins, i.e. we have

$$\text{pay}_{\{x, x'\}} = \text{pay}_{\{x, x'\}}^{\circ}.$$

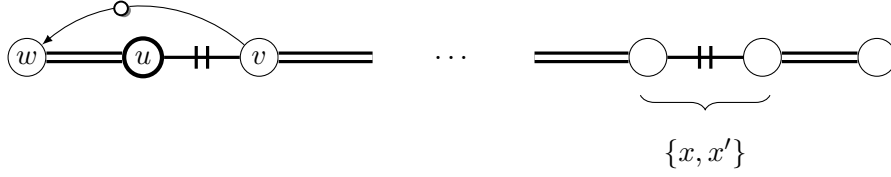


Figure 16: Case 2: The 1-2-GREEDY algorithm picks edge $\{x, x'\}$ in the second step for an endpoint of X ; which of x and x' is the M^* -neighbor of the endpoint of X drawn right depends on the sub-case (not all edges are drawn)

We show that x and x' pay at most $\text{pay}_{\{x, x'\}}^\circ \leq \text{PAY}_{\{x, x'\}} - 1$ coins. Assume the opposite, i.e. that $\text{pay}_{\{x, x'\}}^\circ = \text{PAY}_{\{x, x'\}}$ holds. We show a contradiction.

First, we argue that a degree-1 endpoint exists after edge $\{x, x'\}$ is picked. By Lemma 29, we have $\text{pay}_{\{x, x'\}}^\circ = \text{PAY}_{\{x, x'\}} = 2(\Delta - 2)$. Thus each of x and x' pays $\text{pay}_x^\circ = \text{pay}_{x'}^\circ = \Delta - 2$ coins, since each is incident with at most $\Delta - 2$ edges of F . In particular, each of x and x' is incident with exactly $\Delta - 2$ transfer F -edges in step s , by Definition 10 (Transfer). Hence each of x and x' has degree at least $\Delta - 1$ in step s , since it is also incident with M -edge $\{x, x'\}$. In particular, the 1-2-GREEDY algorithm selects x with degree $d_s(x) \geq \Delta - 1 \geq 3$ (recall that we assume $\Delta \geq 4$).

Note that $\text{pay}_{\{x, x'\}}^\circ = 2(\Delta - 2) \geq 1$ holds and hence a transfer leaves x or x' , say to endpoint w' . By Definition 10 (Transfer), after step s endpoint w' has degree at most $d_{s+1}(w') \leq 1$. In particular, endpoint w' has degree exactly $d_{s+1}(w') = 1$, since during step s the degree of w' drops by at most two from at least $d_s(w') \geq d_s(x) \geq 3$.²²

A degree-1 node other than w' is selected in step $s + 1$, call it y . But node y had degree at least $d_s(y) \geq d_s(x) \geq 3$ in step s as well, consequently edges $\{x, y\}$ and $\{x', y\}$ are removed from the graph during step s . Both $\{x, y\}$ and $\{x', y\}$ are not transfers, since x, x' , and y are M -matched. Now observe that at least one of $\{x, y\}$ and $\{x', y\}$ is an F -edge: at most one of both edges can be an M^* -edge, since otherwise two M^* -edges would be incident with y . Say $\{x, y\}$ is an F -edge. Since $\{x, y\}$ is not a transfer, node x saves a coin and we get $\text{pay}_{\{x, x'\}}^\circ \leq \text{PAY}_{\{x, x'\}} - 1$, the desired contradiction.

Case 2.2: Static Donation. Now assume that x pays a static donation. Recall that, by Definition 26 (Donation), in this static donation we move $\text{pay}_x^\square = \Delta - 3$ bills away from X . By Lemma 27 a), node x pays $\text{pay}_x^\circ = 0$ coins. Furthermore, node x'

²²Here we make use of $\Delta \geq 4$. In particular, our analysis does not work for $\Delta = 3$, since in this case we cannot rely on the existence of a degree-1 endpoint in step $s + 1$.

pays $\text{pay}_{x'}^{\square} = 0$ bills, by definition of donations, and at most $\text{pay}_{x'}^{\circ} \leq \text{PAY}_{x'}^{\circ} = \Delta - 2$ coins, by Lemma 12 b). Therefore we get $\text{pay}_{\{x,x'\}} \leq (\Delta - 3) + (\Delta - 2) = 2(\Delta - 2) - 1 = \text{PAY}_{\{x,x'\}} - 1$.

Case 2.3: Dynamic Donation. Lastly, assume that x pays a dynamic donation (x, \bar{v}) to node \bar{v} . By Definition 26 (Donation), node \bar{v} belongs to a path $\bar{X} \neq X$ for which a degree-1 endpoint exists after creation. Moreover, when \bar{v} becomes matched, say in step \bar{t} , the 1-2-GREEDY algorithm selects a node \bar{u} with degree $d_{\bar{t}}(\bar{u}) \geq 3$ to create a path \bar{X} , and in step $s = \bar{t} + 1$ the 1-2-GREEDY algorithm selects x with degree $d_{\bar{t}+1}(x) = 1$. Therefore x is not incident its M^* -edge and the situation is as depicted in Figure 17.

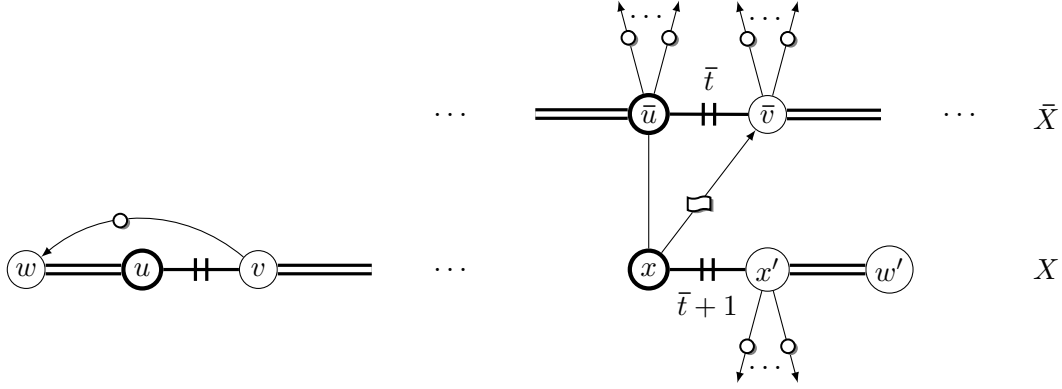


Figure 17: Case 2.3: The 1-2-GREEDY algorithm picks node x with degree $d_{\bar{t}+1}(x) = 1$ and node x' is the M^* -neighbor of an endpoint of X (not all edges are drawn)

First, we bound the payments of nodes x and x' . Assume that in step \bar{t} node \bar{u} becomes matched with \bar{v} , and let sets $W, W_1, W_1^1, W_1^2, W_2$, and $W_{\geq 3}$ be sets of endpoints neighboring \bar{u} and \bar{v} as defined in Definition 32. By Lemma 31₃, nodes \bar{u} and \bar{v} pay $\text{pay}_{\{\bar{u},\bar{v}\}}^{\circ} = 2|W_1^2| + |W_1^1|$ coins and node x' pays at most $\text{pay}_{x'}^{\circ} \leq |W_1^1| + |W_2|$ coins. Now recall that along (x, \bar{v}) we move exactly $\text{pay}_x^{\square} = \text{pay}_{\{\bar{u},\bar{v}\}}^{\circ}$ bills to \bar{v} . Hence x and x' pay at most $\text{pay}_{\{x,x'\}} = \text{pay}_x^{\square} + \text{pay}_{x'}^{\circ} = \text{pay}_{\{\bar{u},\bar{v}\}}^{\circ} + \text{pay}_{x'}^{\circ}$, since we have $\text{pay}_x^{\circ} = 0$ by Lemma 27 a) and $\text{pay}_{x'}^{\square} = 0$ Definition 26 (Donation).

In particular, by Lemma 31₃ we have $\text{pay}_{\{x,x'\}} \leq 2(\Delta - 2) = \text{PAY}_{\{x,x'\}}$. If $\text{pay}_{\{x,x'\}} \leq \text{PAY}_{\{x,x'\}} - 1$ holds then we are done.

So assume from here on that x and x' pay $\text{pay}_{\{x,x'\}} = \text{PAY}_{\{x,x'\}} = 2(\Delta - 2)$ coins and bills. Next, we identify the endpoints which receive a coin from x' . As a consequence of Lemma 31₃ we obtain the following equality:

$$\text{pay}_{\{x,x'\}} = \text{pay}_{\{\bar{u},\bar{v}\}}^{\circ} + \text{pay}_{x'}^{\circ} = 2|W_1^2| + 2|W_1^1| + |W_2| = |\mathcal{E}(W)| = 2(\Delta - 2) = \text{PAY}_{\{x,x'\}}.$$

Since $\text{pay}_{\{\bar{u},\bar{v}\}}^{\circ} = 2|W_1^2| + |W_1^1|$ holds, we get that x' pays exactly

$$\text{pay}_{x'}^{\circ} = |W_1^1| + |W_2|$$

coins. But which endpoints receive coins from x' ? We argue that each endpoint in W_1^1 and W_2 receives exactly one transfer from x' . Therefore we first determine which endpoints do *not* receive transfers from x' .

- Node x' pays zero coin to an endpoint $w \in (\mathcal{W} \setminus W) \cup W_{\geq 3}$.

Why? In step $\bar{t} + 1$ when 1-2-GREEDY picks edge $\{x, x'\}$ endpoint w has degree at least $d_{\bar{t}+1}(w) \geq 3$. Since w is not adjacent to x when x is selected with degree $d_{\bar{t}+1}(x) = 1$, the degrees of w drops by at most one and to at least $d_{\bar{t}+2}(w) \geq 2$. Hence w does not receive a transfer from x' .

Therefore x' might only pay coins to endpoints in $W_2 \cup W_1$.

- But node x' also pays no coins to nodes in W_1^2 .

Why? Since in the step when 1-2-GREEDY picks edge $\{x, x'\}$ each endpoint in W_1^2 already receives two coins from nodes \bar{u} and \bar{v} and any further coin from x' is canceled by Definition 24.

Consequently, node x' might pay coins only to nodes in W_2 or W_1^1 .

Since sets $\mathcal{W} \setminus W$, $W_{\geq 3}$, W_2 , W_1^2 , and W_1^1 are pairwise disjoint and since $\text{pay}_{x'}^{\circ} = |W_1^1| + |W_2|$ holds, we obtain that x' pays exactly one coin to each endpoint in W_1^1 as well as to each endpoint in W_2 .

In the rest of the proof we proceed as follows. We show that either an endpoint of X receives at least two coins, which proves $\text{rcv}_X^{\circ} \geq \text{RCV}_X^{\circ} + 1$, or we show a contradiction to our assumption that nodes x and x' pay $\text{pay}_{\{x,x'\}} = \text{PAY}_{\{x,x'\}}$ coins and bills, which proves $\text{pay}_{\{x,x'\}} \leq \text{PAY}_{\{x,x'\}} - 1$.

We prepare the argument. Consider step $\bar{t} + 1$ when 1-2-GREEDY picks edge $\{x, x'\}$. Recall that x has degree $d_{\bar{t}+1}(x) = 1$ and observe that the M^* -edge of x is already removed from the graph. Recall also that the M^* -edge

$$\{x', w'\}$$

connecting x' with a path endpoint w' of X is removed in this step.

Consider again the path \bar{X} receiving the donation from x , and the nodes \bar{u} and \bar{v} matched to create \bar{X} . We conduct a case analysis based on the type of endpoint w' , i.e. if w' belongs to set $\mathcal{W} \setminus W, W_{\geq 3}, W_2, W_1^2$, or W_1^1 (recall that above we defined these sets for nodes \bar{u} and \bar{v} matched to create path \bar{X}). The 1-2-GREEDY algorithm creates path \bar{X} in step \bar{t} when nodes \bar{u} and \bar{v} have degree at least $d_{\bar{t}}(\bar{u}) \geq 3$ resp. $d_{\bar{t}}(\bar{v}) \geq 3$. Since x pays a donation to \bar{v} , after creation of \bar{X} the degree of x is exactly $d_{\bar{t}+1}(x) = 1$. Since $d_{\bar{t}}(x) \geq d_{\bar{t}}(\bar{u}) \geq 3$ holds in step \bar{t} , two edges $\{\bar{u}, x\}$ and $\{\bar{v}, x\}$ are removed from the graph in the creation step of \bar{X} .

$w' \in (\mathcal{W} \setminus W) \cup W_{\geq 3}$: We show that w' receives at least two coins due to its large degree after creation of path \bar{X} .

In step \bar{t} the degree of w' is at least $d_{\bar{t}}(w') \geq d_{\bar{t}}(u) \geq 3$. If we have $w' \in \mathcal{W} \setminus W$, then w' is not adjacent to \bar{u} or \bar{v} and hence no edges incident with w' are removed in the creation step of \bar{X} . Thus the degree of w' is at least $d_{\bar{t}+1}(w') \geq 3$ after creation of \bar{X} . If $w' \in W_{\geq 3}$ holds, then the degree of w' is at least $d_{\bar{t}+1}(w') \geq 3$ after creation of \bar{X} by Definition 32.

In step $\bar{t} + 1$, when 1-2-GREEDY picks edge $\{x, x'\}$, the degree of endpoint w' drops by at most one, since w' is not adjacent with node x , which is selected with degree $d_{\bar{t}+1}(x) = 1$.

Consider the step after edge $\{x, x'\}$ is picked, i.e. step $\bar{t} + 2$. The degree of w' is at least $d_{\bar{t}+2}(w') \geq 2$ and the M^* -edge of w' is removed from the graph. Hence w' is incident with at least two F -edges. Now observe that w' eventually receives at least two coins, namely over the two F -edges which are removed last from the graph. Consequently, path X receives at least $\text{rcv}_X^\circ \geq \text{RCV}_X^\circ + 1$ coins.

$w' \in W_1^1$: We show a contradiction to our assumption that $\text{pay}_{\{x, x'\}} = \text{pay}_{\{\bar{u}, \bar{v}\}}^\circ + \text{pay}_{x'}^\circ = 2(\Delta - 2)$ holds.

Consider step \bar{t} when 1-2-GREEDY creates path \bar{X} . By definition of W_1^1 (cf. Definition 32), two edges incident with w' are removed during step \bar{t} , since w' has degree at least $d_{\bar{t}}(w') \geq 3$ before and degree exactly $d_{\bar{t}+1}(w') = 1$ afterwards. Also by definition of W_1^1 , only one F -edge incident with w' is removed, hence the M^* -edge incident with w' is removed as well. Since the M^* -edge of w' is removed in the creation step of \bar{X} , endpoint w' belongs to \bar{X} .

Thus w' belongs to each of paths X and \bar{X} . A contradiction, since $X \neq \bar{X}$ holds by definition of (x, \bar{v}) .

$w' \in W_1^2$: By definition of set W_1^2 , two F -edges e_1 and e_2 incident with w' are removed when path \bar{X} is created in step \bar{t} and after creation of \bar{X} the degree of w' is exactly $d_{\bar{t}+1}(w') = 1$. In particular, edges e_1 and e_2 are transfers to w' . But since both edges are removed from the graph in the same step, both transfers are never canceled by Definition 24. Hence w' receives at least two coins, and path X receives at least $\text{rev}_X^\circ \geq \text{RCV}_X^\circ + 1$ coins.

$w' \in W_2$: Recall that $\text{pay}_{x'}^\circ = |W_1^1| + |W_2|$ holds and that node x' pays a coin to each endpoint in W_1^1 and to each endpoint in W_2 . By the assumption $w' \in W_2$, node x' pays a coin to w' , namely in transfer (x', w') . Since only F -edges can be transfers, we have $\{x', w'\} \in F$. A contradiction, since $\{x', w'\} \in M^*$ holds. \square

4 Performance Guarantees for KARPSSIPER

For KARPSSIPER on bipartite graphs we prove a stronger performance guarantee than for 1-2-GREEDY on general graphs, cf. Theorems 15 and 21.

Theorem 35. *The KARPSSIPER algorithm achieves approximation ratio at least $\frac{\Delta}{2\Delta-2}$ for each bipartite graph of degree at most Δ , for $\Delta \geq 3$.*

As in the analysis of 1-2-GREEDY in the proof we study the graph $H(\text{KARPSSIPER})$ whose connected components are paths and singletons (w.l.o.g. we may assume that other component types do not exist, cf. Lemma 4 and Section 2.3). However, whereas the graph $H(1\text{-}2\text{-GREEDY}) = (V, M \cup M^*)$ is defined on node set V , here the graph is defined on set $L \cup R$ of nodes in the left resp. right partition, i.e. we have

$$H(\text{KARPSSIPER}) = (L \cup R, M \cup M^*).$$

A fundamental difference to general graphs is that degrees drop by at most one in each step, since a bipartite graph does not contain triangles.

Organization of the Proof. To verify Theorem 35, by Lemma 7 it suffices to show that all paths and singletons have local approximation ratio at least $\frac{\Delta}{2\Delta-2}$. Therefore we present sufficient balance bounds in Section 4.1, where we also outline similarities and differences to the analysis of 1-2-GREEDY. In Section 4.2 we sketch the proof for paths. In Section 4.3 we present details of the proofs for paths and singletons.

4.1 Overview of Modifications

We proceed as in the proof for 1-2-GREEDY and bound the number of coins and bills owned by each component, i.e. we bound *balances*, cf. Definition 5. However, the minimum balances required by (5 Singleton) and (6 Path) for 1-2-GREEDY are insufficient to obtain the claimed $\frac{\Delta}{2\Delta-2}$ -guarantee for KARPSSIPER, as follows from the $\frac{\Delta-1}{2\Delta-3}$ -inapproximability bound given in Theorem 64 in the second part of the thesis.

Modified Balance Bounds. For singletons we reuse bound (5 Singleton) as is: a singleton X has to have balance at least $\text{bal}_X \geq \text{BAL}_X = -2(\Delta - 2)$. However, we slightly increase the required balance for a path X to

$$\text{BAL}_X = \underbrace{2(\Delta - 1) - 2m_X(\Delta - 2)}_{(6 \text{ Path})} + 2 = 2\Delta - 2m_X(\Delta - 2). \quad (6 \text{ Path}_{\text{KS}})$$

Lemma 36. *If bounds (5 Singleton) and (6 Path_{KS}) hold for all singletons resp. paths, then there is a coin and bill value κ such that all components are $\frac{\Delta}{2\Delta-2}$ -balanced.*

Proof. First, we bound local approximation ratios of singletons and paths parameterized by κ . Then we choose κ such that both bounds are at least $\frac{\Delta}{2\Delta-2}$.

Using (5 Singleton) for a singleton X , i.e. that the balance of X is at least $\text{bal}_X \geq \text{BAL}_X = -2(\Delta - 2)$, the local approximation ratio of X is at least

$$\frac{1 + \kappa \cdot \text{BAL}_X}{1} \geq 1 - \kappa 2(\Delta - 2).$$

By (6 Path_{KS}) the balance of a path X is at least $\text{bal}_X \geq \text{BAL}_X = 2\Delta - 2m_X \cdot (\Delta - 2)$. Hence the local approximation of X is at least

$$\begin{aligned} \frac{m_X + \kappa \cdot \text{BAL}_X}{m_X + 1} &\geq \frac{m_X + \kappa(-2m_X(\Delta - 2) + 2\Delta)}{m_X + 1} \\ &= \frac{m_X \cdot (1 - \kappa 2(\Delta - 2)) + \kappa 2\Delta}{m_X + 1} \\ &= \frac{(m_X + 1) \cdot (1 - \kappa 2(\Delta - 2)) - (1 - \kappa 2(\Delta - 2)) + \kappa 2\Delta}{m_X + 1} \\ &= 1 - \kappa 2(\Delta - 2) + \frac{\kappa 2(2\Delta - 2) - 1}{m_X + 1}. \end{aligned} \quad (12)$$

We choose $\kappa = \frac{1}{2(2\Delta-2)}$. Consequently, bound (12) can be rewritten as $1 - \kappa 2(\Delta - 2)$. Now observe that the bounds for both singletons and paths become

$$1 - \kappa 2(\Delta - 2) = \frac{2\Delta}{2(2\Delta - 2)} = \frac{\Delta}{2\Delta - 2}. \quad \square$$

Modifications. The modified balance bounds cannot be verified without a revision of the proof method applied for 1-2-GREEDY. We proceed with a rough overview of modifications.

Unified Case Analysis. Recall that by Proposition 2 both 1-2-GREEDY and the KARPSIPSER algorithm perform optimally if in the input graph any degree is bounded by at most two. Therefore we restrict the proof to maximum degree Δ , for $\Delta \geq 3$.

1-2-GREEDY: Here we investigate the cases $\Delta = 3$ and $\Delta \geq 4$ separately. For $\Delta = 3$ our analysis depends only on the use of transfers but not on donations.

KARPSIPSER: The proof for $\Delta = 3$ cannot be carried out using transfers only, as we demonstrate in Example 37 below. Therefore we prove cases $\Delta = 3$ and $\Delta \geq 4$ simultaneously, using both transfers *and* donations.

Example 37. Consider the graph depicted in Figure 18. All nodes have degree at most $\Delta = 3$. The KARPSIPSER algorithm creates the $\frac{1}{2}$ -path X in the first step (when no degree-1 node exists). In the second step, KARPSIPSER creates a singleton by selecting the bold degree-1 node. Similarly, KARPSIPSER creates another $\frac{1}{2}$ -path in step three and another singleton in step four. In the end, KARPSIPSER creates three more singletons in steps five, six, and seven, each time selecting a degree-1 node. A coin is moved along each indicated transfer.

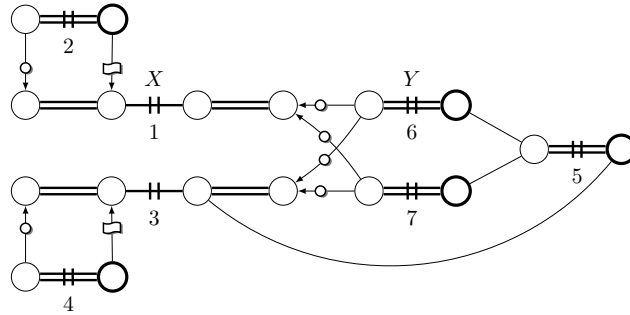


Figure 18: Transfers allow no tight approximation guarantee for $\Delta = 3$ (KARPSIPSER picks edges in the order indicated by numbers next to M -edges, where selected degree-1 nodes are drawn bold)

To demonstrate that transfers are insufficient to prove approximation ratio at least $\frac{\Delta}{2\Delta-2} = \frac{3}{4}$, ignore the two donations indicated by bills in the figure. We argue that there does not exist a coin value κ such that all local approximation ratios are at least $\frac{3}{4}$. Why? Path X receives three coins in total. To obtain local approximation ratio at least $\frac{1+3\kappa}{2} \geq \frac{3}{4}$ for X , we have to choose $\kappa \geq \frac{1}{6}$. On the other hand, singleton Y pays two coins. To obtain local approximation ratio at least $\frac{1-2\kappa}{1} \geq \frac{3}{4}$ for Y , we have to choose $\kappa \leq \frac{1}{8}$. This contradicts $\kappa \geq \frac{1}{6}$.

New Types of Donations. We develop a new system of donations, since Definition 26 cannot be applied to bipartite graphs. Why?

1-2-GREEDY: Recall that a dynamic donation, cf. Definition 26, is initiated when a node of degree at least three is selected in a path creation step and a degree-1

endpoint exists in the subsequent step. In particular, the minimum degree drops from at least three to one.

KARPSIPSER: In a bipartite graph, however, in each step degrees drop by at most one since the graph does not contain triangles.

As for 1-2-GREEDY, a source node of (one type of) a donation becomes matched after creation of a path when it has degree one.

Example 38. *In Figure 18, the $\frac{1}{2}$ -path X receives a donation. In particular, in this donation we move one bill. Therefore the local approximation ratio of X is now $\frac{1+(3+1)\kappa}{2}$ and a coin and bill value of $\kappa = \frac{1}{8}$ is now sufficient for both X and singleton Y to be $\frac{3}{4}$ -balanced.*

Compared with 1-2-GREEDY, the most prominent difference of donations for KARPSIPSER is the following. The maximum number of bills that we move in a donation is smaller than for 1-2-GREEDY, namely at most

$$\Delta - 2$$

instead of up to $2(\Delta - 2)$. However, in order to obtain sufficient funds, a path might now receive up to two donations, which is not the case for 1-2-GREEDY.

We formally define the new donation types in Definition 41 and thereafter give a detailed comparison with 1-2-GREEDY in Table 1.

No Transfer Cancellations. In the analysis of 1-2-GREEDY we cancel certain transfers as per Definition 24. For bipartite graphs, Definition 24 is not applicable. Why?

1-2-GREEDY: A path endpoint w receives up to three transfers by Lemma 23. If w receives three transfers, then by Definition 24 we cancel the “third” incoming transfer, which is uniquely defined.

KARPSIPSER: On the other hand, the following result shows that for bipartite graphs each path endpoint receives at most two transfers: there is no need to cancel a third transfer. Hence, in the analysis of KARPSIPSER we use transfers exactly as defined in Definition 10.

Lemma 39. *In a bipartite graph, a path endpoint w receives $1 \leq \text{rcv}_w^{\circ} \leq 2$ coins. A path X receives at least $\text{rcv}_X^{\circ} \geq 2$ coins.*

Proof. By Lemma 13 we have $\text{rcv}_w^\circ \geq 1$. As a direct consequence we get $\text{rcv}_X^\circ \geq 2$ and it remains to show that $\text{rcv}_w^\circ \leq 2$ holds. By Definition 10, an F -edge $\{v, w\}$ becomes a transfer to w when v becomes matched *and* the degree of w drops below two. Initially, the degree of w is $d_1(w) \geq 2$ by Lemma 8 b) and it drops by at most one in each step, since the graph is bipartite. Thus the degree of w drops to below two in exactly two steps, namely when it drops to one and then to zero. \square

4.2 Paths

To motivate the approach, we first determine an upper bound on the funds paid by a path X , namely in (14) below. As for 1-2-GREEDY, the following two facts hold. First, path endpoints of X do not pay funds: a bound on the payments of X is implied from a bound on the payments of an M -covered node x of X . Secondly, if node x pays a donation then x becomes matched with degree one and consequently x does not pay a transfer: node x pays coins or bills, but not both.

In particular, we have $\text{pay}_x \leq \max\{\text{pay}_x^\circ, \text{pay}_x^\square\}$. Observe that $\text{pay}_x^\circ \leq \text{PAY}_x^\circ = \Delta - 2$ holds by Lemma 12 b). By Definition 41, at most one donation leaves x in which we move at most $\text{pay}_x^\square \leq \text{PAY}_x^\square = \Delta - 2$ bills. Consequently, each path node pays at most

$$\text{PAY}_x = \text{PAY}_x^\circ = \text{PAY}_x^\square = \Delta - 2 \quad (13)$$

coins or bills and we obtain the same upper bounds on edge payments and path payments as for 1-2-GREEDY, cf. Lemma 29, namely (denote the M -neighbor of x as x')

$$\begin{aligned} \text{PAY}_{\{x, x'\}} &= 2(\Delta - 2) & \text{and} \\ \text{PAY}_X &= m_X \cdot 2(\Delta - 2). \end{aligned} \quad (14)$$

Using (14) we verify (6 Path_{KS}), namely that $\text{bal}_X = \text{rcv}_X - \text{pay}_X \geq \text{BAL}_X = 2\Delta - m_X \cdot 2(\Delta - 2)$ holds, as follows. We prove that “savings” and “revenues” of nodes of X —which are not regarded in (14)—add up to at least

$$2\Delta.$$

Therefore we identify two sets I_L and I_R of nodes such that savings and revenues of each set add up to at least Δ . Since these nodes increase the “wealth” of path X , we call nodes in I_L and I_R

left increase nodes resp. *right increase nodes*

of X . To prove a total increase of at least 2Δ for both sets we show that $I_L \cap I_R = \emptyset$ holds, see Lemma 49 in Section 4.3.

Sets I_L and I_R are subsets of gray resp. white nodes in Figure 19. (We do not formally define both sets until Table 2 below, since the definition requires further preparation, which is provided throughout the rest of Section 4.2.) Path X has odd length, hence a path endpoint of X is contained in each of partitions L and R , call them w_L resp. w_R . Nodes matched in the step for endpoint w_R , cf. Section 2.1.1, are denoted as x'_R and x_L , where we assume that $\{w_R, x_L\} \in M^*$ holds. Nodes at the other end of X are defined analogously. We denote the nodes matched by KARPSIPSER in the creation step of X as u_L and u_R , where we assume $u_L \in L$ resp. $u_R \in R$. (Note that we have $\{u_L, u_R\} = \{x_L, x'_R\}$ if X is created in the step for endpoint w_R , that we have $\{u_L, u_R\} = \{x'_L, x_R\}$ if X is created in the step for w_L , and that we have $\{u_L, u_R\} = \{x_L, x'_R\} = \{x'_L, x_R\}$ if X is a $\frac{1}{2}$ -path.) Nodes v_R and v_L are M -matched neighbors of u_L resp. u_R (with $v_R \neq u_R$ and $v_L \neq u_L$) and each of v_R and v_L might belong to X , to another path, or to a singleton; the precise definition of v_R and v_L is found after some more preparation in Definition 40.

In the rest of Section 4.2 we primarily focus on increase node in I_L since the argument for I_R is symmetric.

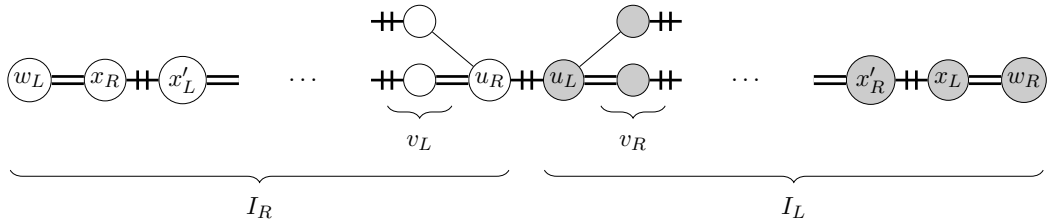


Figure 19: Naming convention for increase nodes (nodes v_L and v_R are chosen as one of the indicated nodes)

4.2.1 New Types of Donations

As for 1-2-GREEDY, in a donation we move bills to a node becoming matched in the creation step of a path, e.g. node u_L . Again, the basic idea is to move enough bills in order to compensate for all coins paid by u_L . Node v_R plays a central role in the argument.

“ $v_R \notin X$ ”: Either v_R belongs to another component $Y \neq X$ than u_L . Then in a donation from v_R to u_L we move $\text{pay}_{v_R}^{\square} = \text{rcv}_{u_L}^{\square}$ bills to X such that

$$\text{rcv}_{u_L}^{\square} \geq \text{pay}_{u_L}^{\circ}$$

holds. We motivate, define, and discuss donations in the rest of this section. In particular, the given “compensation inequality” is verified in Lemma 43.

“ $v_R \in X$ ”: Or v_R is also a node of X . In this case we show that v_R saves many coins for X . We analyze this case later in Section 4.3.

Towards the Definition. First, we discuss crucial differences between the donation types for 1-2-GREEDY and KARPSSIPER. For 1-2-GREEDY we initiate a donation only if a node (matched to create a path Z) has to pay coins and for the following “trigger”, cf. Definition 26:

we demand that in the step after creation of Z there exists a degree-1 path endpoint.

For KARPSSIPER, the definition of a donation depends on this trigger as well, but *not* on whether coins have to be payed, cf. Definition 41 below.

(We note that if u_L pays coins, then the trigger occurs. Why? Recall that nodes u_L and u_R have degree at least two in the creation step t of path X —this is a consequence of Lemma 8 a). By definition of KARPSSIPER we have $d_t(w) \geq 2$ also for each endpoint w , in particular for an endpoint w receiving a coin from u_L . By Definition 10 (Transfer), endpoint w has degree $d_{t+1}(w) \leq 1$ in the subsequent step. Now observe that degrees drop by *at most one* in each step, since the graph is bipartite. Consequently, endpoint w has degree *exactly* $d_{t+1}(w) = 1$ after creation of X : a degree-1 endpoint exists.)

If the trigger occurs, then our plan is to initiate a donation to u_L . The second crucial difference is the definition of a donation source node. For 1-2-GREEDY, such a node becomes matched with degree one in the step following the creation of the receiving path. This is also the case for KARPSSIPER, but only for the source node of the “first” donation received by a path. The path might receive one additional donation, whose source node might become matched even later, but also with degree one. After the following definitions we argue in Lemma 42 that the given choice of node v_R is valid, i.e. that v_R really exists.

Definition 40 (The Node v_R). *Let X be a path created in step t when KARPSIPSER matches node u_L with degree $d_t(u_L) \geq 2$. Assume that in step $t + 1$ partition R contains an endpoint w of degree $d_{t+1}(w) = 1$.*

Let $R_1 = \{v \in R : d_{t+1}(v) = 1, \{u_L, v\} \in E\}$ be the set of all degree-1 nodes in R in step $t + 1$. We choose v_R as the node in R_1 which becomes matched earliest. (Note that KARPSIPSER does not necessarily select v_R .)

Definition 41 (Donation_{KS}). *Let X be a path created in step t when KARPSIPSER matches node u_L with degree $d_t(u_L) \geq 2$. Assume that in step $t + 1$ partition R contains an endpoint w of degree $d_{t+1}(w) = 1$.*

If node v_R belongs to another component $Y \neq X$ than u_L , then edge $\{u_L, v_R\} \in F$ is called a donation and is denoted as (v_R, u_L) . We move bills from v_R to u_L as follows.

- *We move $\Delta - 3$ bills and call (v_R, u_L) a small donation unless the following holds.*
- *When v_R becomes matched, partition R contains exactly $\Delta - 1$ degree-1 nodes, namely v_R and $\Delta - 2$ degree-1 path endpoints; in this case we move $\Delta - 2$ coins and call (v_R, u_L) a large donation.*

The maximum number of bills payed by an M -matched node x is $\text{PAY}_x^\square = \Delta - 2$.

To see that v_R exists, recall that we demand that there is a degree-1 endpoint in R after creation of X and apply the following result for $s = t + 1$.

Lemma 42. *Consider some step s and assume that R contains a degree-1 path endpoint w , i.e. we have $d_s(w) = 1$. Then R also contains an M -matched node v with $d_s(v) = 1$.*

Proof. Assume that in step s all degree-1 nodes in partition R are path endpoints, one of which is w . Since path endpoints are never matched, an edge with a degree-1 node u in partition L is picked next, say u becomes matched with u' . Observe that u' is an M -matched node in R and that all degrees in partition R , but that of u' , are not changed when edge $\{u, u'\}$ is picked. Hence the set of degree-1 endpoints in R and their degrees remain unchanged. By repeating the argument we get that the degree of w is never decreased to zero. A contradiction. \square

Similarities and Differences. In Table 1 we summarize similarities and differences of donations used in the analysis of 1-2-GREEDY resp. KARPSIPSER. We comment on the rows of Table 1 in turn. Let (y, x) be an “old” or “new” donation.

Table 1: Properties of a donation (y, x) to M -edge $\{x, x'\}$ (differences are marked gray, by $t(z)$ we denote the step when the algorithm matches node z)

	Property	1-2-GREEDY	KARPSIPSER
1	$\{y, x\}$ is a transfer	no	no
2	$\{y, x\} \in F$	yes	yes
3	A path is created in step $t(x)$	yes	yes
4	Node y becomes matched in step	$t(y) = t(x) + 1$	$t(y) > t(x)$
5	Degree of y in step $t(y)$	1	1
6	Number pay_y° of coins payed by y	0	0
7	Number $\text{pay}_y^\square = \text{rcv}_x^\square$ of bills moved to x	$\leq 2(\Delta - 2)$	$\leq \Delta - 2$
8	Number of donations received per path	≤ 1	≤ 2
9	Number of donations payed per M -edge	≤ 1	≤ 2
10	Node y pays at most one donation	yes	yes
11	Node x receives at most one donation	yes	yes
12	Donated bills compensate payed coins	$\text{rcv}_{\{x, x'\}}^\square \geq \text{pay}_{\{x, x'\}}^\circ$	$\text{rcv}_x^\square \geq \text{pay}_x^\circ$

1. Both y and x are M -matched nodes, hence (y, x) is not a transfer.
- 2/3. Donation (y, x) is an F -edge and the destination node x becomes matched to create a path X .
4. The source node y becomes matched later, namely in the subsequent step for 1-2-GREEDY. For KARPSIPSER, the source node of at least one donation to X (if any) also becomes matched in the step after creation; the source node of an additional donation to X (if any) might become matched even later, cf. left example in Figure 20.
- 5/6. For both algorithms node y becomes matched with degree one, therefore no transfer leaves y and $\text{pay}_y^\circ = 0$ holds.
7. New donation types move smaller amounts of bills, namely at most $\Delta - 2$ instead of up to $2(\Delta - 2)$.
8. However, for KARPSIPSER a path receives up to two donations instead of at most one. Thus the maximum number of bills received per path is unchanged.
9. For KARPSIPSER, if source nodes of two donations to a path are matched in the same step, then the according M -edge pays two donations, cf. right example in Figure 20.

10/11. In any case, each node pays and receives at most one donation.

12. For both algorithms, in a donation we move enough bills to compensate for all coins paid by the “receiver”: for 1-2-GREEDY the coins paid by the receiving M -edge of x are compensated by Lemma 27 c), for KARPSIPSER the coins paid by the receiving node x are compensated by the following result.

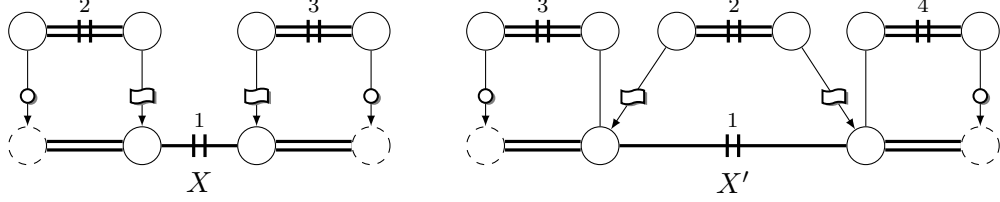


Figure 20: Paths X and X' receive two donations, where for X' both donations come from the same M -edge (small numbers indicate steps in which KARPSIPSER picks an edge, dashed endpoints have degree one after creation of their respective paths)

Lemma 43. *If node u_L receives a donation (v_R, u_L) , then $\text{rcv}_{u_L}^{\square} \geq \text{pay}_{u_L}^{\circ}$ holds.*

Proof. Since v_R is an M -matched node, the F -edge $\{v_R, u_L\}$ is not a transfer. Consequently, node u_L pays at most $\Delta - 3$ transfers and we get $\text{pay}_{u_L}^{\circ} \leq \Delta - 3$. Now observe that in (v_R, u_L) , whether (v_R, u_L) is a small or a large donation, we move at least $\text{pay}_{v_R}^{\square} = \text{rcv}_{u_L}^{\square} \geq \Delta - 3$ bills to u_L . \square

Revenues and Savings of u_L . For the analysis of KARPSIPSER, we need fine-grained control over revenues and savings of a path. Therefore we introduce the following convenient notation for savings. Recall that an M -covered node x does not pay coins and bills at the same time and that $\text{PAY}_X = \text{PAY}_x^{\circ} = \text{PAY}_x^{\square} = \Delta - 2$ is an upper bound on the number of coins resp. bills paid by x , cf. (13).

Definition 44. *For an M -matched node x we denote by sav_x° , sav_x^{\square} , and sav_x the number of coins, bills, resp. funds saved by x . In particular, we have*

$$\begin{aligned} \text{sav}_x &= \text{PAY}_x = (\Delta - 2) && \text{if } \text{pay}_x^{\circ} = \text{pay}_x^{\square} = 0, \\ \text{sav}_x = \text{sav}_x^{\circ} = \text{PAY}_x^{\circ} - \text{pay}_x^{\circ} &= (\Delta - 2) - \text{pay}_x^{\circ} && \text{if } \text{pay}_x^{\circ} > \text{pay}_x^{\square} = 0, \text{ and} \\ \text{sav}_x = \text{sav}_x^{\square} = \text{PAY}_x^{\square} - \text{pay}_x^{\square} &= (\Delta - 2) - \text{pay}_x^{\square} && \text{if } \text{pay}_x^{\square} > \text{pay}_x^{\circ} = 0. \end{aligned}$$

Lemma 45. *If node u_L pays $\text{pay}_{u_L}^\circ > 0$ coins and receives a donation, then revenues and savings of u_L are bounded by at least*

$$\text{rcv}_{u_L}^\square + \text{sav}_{u_L}^\circ \geq \Delta - 2.$$

Proof. By Lemma 43 and Definition 44 we get $\text{rcv}_{u_L}^\square + \text{sav}_{u_L}^\circ \geq \text{pay}_{u_L}^\circ + \text{sav}_{u_L}^\circ = \Delta - 2$. \square

In particular, since bound (14) on the payments of path X does not incorporate any savings or revenues, we obtain that the balance of X is by at least $\Delta - 2$ larger than (14).

4.2.2 Steps for Path Endpoints

Recall that our goal is to show that savings and revenues of left increase nodes add up to at least Δ . By Lemma 45, it suffices to show additional revenues and savings of at least two. Therefore we study endpoint w_R as well as the nodes x_L and x'_R becoming matched in the step for w_R .

First, recall that by Lemma 39 endpoint w_R receives either $\text{rcv}_{w_R}^\circ = 1$ or $\text{rcv}_{w_R}^\circ = 2$ coins. We refine Lemma 39 in the following result, where we precisely characterize the conditions that w_R receives one or two coins.

Lemma 46. *Consider path X and assume that step s is the step for endpoint w_R of X , i.e. when nodes x_L and x'_R of X become matched. Endpoint w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins if and only if w_R has degree at least $d_{s+1}(w_R) \geq 2$ in the subsequent step (when the M^* -edge of w_R is removed). Otherwise we have $\text{rcv}_{w_R}^\circ = 1$.*

Proof. We prepare the proof with general observations. By Lemma 8 b), endpoint w_R has degree at least $d_1(w_R) \geq 2$ in the input graph G . Since G is bipartite, degrees drop by at most one in each step. Hence there is a step s' when w_R has degree exactly $d_{s'}(w_R) = 2$.

By Definition 10 (Transfer), an edge is not a transfer to w_R if it is removed before step s' . Beginning with step s' , each F -edge removed from w_R is a transfer to w_R .

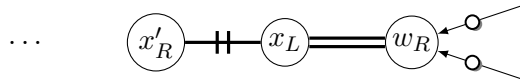


Figure 21: Endpoint w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins (not all edges are drawn)

Assume that $d_{s+1}(w_R) \geq 2$ holds after the step for w_R . Since the M^* -edge $\{w_R, x_L\}$ is already removed from the graph, endpoint w_R is incident with $d_{s+1}(w_R) \geq 2$ many F -

edges. In step $s' \geq s$ endpoint w_R is incident with exactly two F -edges, cf. Figure 21. By the above considerations, both F -edges are transfers. Hence we get $\text{rcv}_{w_R}^\circ = 2$.

Now assume that we have $d_{s+1}(w_R) \leq 1$. The M -edge $\{w_R, x_L\}$ is removed in step s , and since degrees drop by at most one, it is the only edge removed from w_R . Therefore endpoint w_R has at most two incident edges in step s , namely $\{w_R, x_L\}$ and possibly an F -edge. Therefore w_R receives at most one transfer and we get $\text{rcv}_{w_R}^\circ \leq 1$. Since w_R also receives at least one transfer by Lemma 13, we get $\text{rcv}_{w_R}^\circ = 1$. \square

We are now ready to analyze revenues and savings of nodes w_R , x_L , and x'_R involved in the step for endpoint w_R . In particular, in this section we focus on the case that path X is created *before* the step for w_R ; the case that the creation step of X and the step for w_R are the same is analyzed in Section 4.3.

Observe that the next result applies whether or not x'_R pays a donation. Node x_L does not pay a donation, since x_L becomes matched when having an incident M -edge and M^* -edge whereas a donation source node becomes matched with degree one.

Lemma 47. *Assume that the step s for endpoint w_R of path X happens after the creation step of X . We have $\text{rcv}_{w_R}^\circ + \text{sav}_{x_L} + \text{sav}_{x'_R} \geq 2$.*

Proof. Recall that in the step for w_R nodes x_L and x'_R become matched and that x_L does not pay a donation. We distinguish whether a donation leaves node x'_R .

No Donation Leaves x'_R . If w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins, then we are done.

Otherwise w_R receives exactly $\text{rcv}_{w_R}^\circ = 1$ coin, by Lemma 46, which we assume from here on. Recall that we have $\text{pay}_{x_L}^\square = \text{pay}_{x'_R}^\square = 0$, since neither x'_R nor x_L pays a donation, by assumption. We are done if one of x_L and x'_R saves at least one coin.

So assume that none saves a coin, i.e. that $\text{sav}_{x_L}^\circ = \text{sav}_{x'_R}^\circ = 0$ holds. We show a contradiction. Nodes x_L and x'_R pay

$$\begin{aligned} \text{pay}_{x_L}^\circ &= \text{PAY}_{x_L}^\circ - \text{sav}_{x_L}^\circ = \Delta - 2 && \text{resp.} \\ \text{pay}_{x'_R}^\circ &= \text{PAY}_{x'_R}^\circ - \text{sav}_{x'_R}^\circ = \Delta - 2 \end{aligned}$$

coins, since the maximum number of coins paid by a path node x is $\text{PAY}_x^\circ = \Delta - 2$ by Lemma 12 b). Consequently each of x'_R and x_L is incident with $\Delta - 2$ many F -edges in step s , i.e. their degrees are at least $d_s(x'_R) \geq \Delta - 1 \geq 2$ resp. $d_s(x_L) \geq \Delta \geq 3$ in the step for w_R , see Figure 22. Hence, by definition of KARPSIPSER, all degrees in the reduced graph G_s are at least 2.

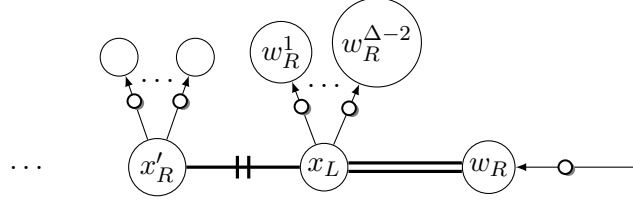


Figure 22: Both x_L and x'_R pay $\Delta - 2$ transfers, endpoint w_R receives one transfer (not all edges are drawn)

After x'_R and x_L are matched, i.e. in step $s + 1$, the destination nodes of transfers from x_L , call them $w_R^1, \dots, w_R^{\Delta-2}$, have degree exactly $d_{s+1}(w_R^1) = \dots = d_{s+1}(w_R^{\Delta-2}) = 1$: they have degree at least two in G_s and most one in step $s + 1$ by Definition 10 (Transfer), and their degrees are decreased by exactly one since the graph is bipartite. Also in step $s + 1$, endpoint w_R has degree exactly $d_{s+1}(w_R) = 1$. Why? First, endpoint w_R has degree at least $d_s(w_R) \geq 2$ in G_s , hence w_R has degree at least $d_{s+1}(w_R) \geq 1$ afterwards. Secondly, to show that the degree is now exactly $d_{s+1}(w_R) = 1$, observe that $d_{s+1}(w_R) \geq 2$ would imply—by Lemma 46—that w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins, which contradicts our assumption.

Consequently, in step $s + 1$ all neighbors of x_L but x'_R , namely w_R and $w_R^1, \dots, w_R^{\Delta-2}$, are now degree-1 endpoints and they are the only degree-1 nodes in R . Thus R contains no M -matched degree-1 node and we obtain a contradiction by Lemma 42.

A Donation (x'_R, y_L) Leaves x'_R . No transfer leaves x'_R and we have $\text{pay}_{x'_R}^\circ = 0$, since x'_R becomes matched with degree one. If (x'_R, y_L) is a small donation and we move $\Delta - 3$ bills, then x'_R saves $\text{sav}_{x'_R}^\square = 1$ bill, by Definition 44. Since endpoint w_R receives at least $\text{rcv}_{w_R}^\circ \geq 1$ coin by Lemma 46, we are done.

Now assume that (x'_R, y_L) is a large donation and we move $\Delta - 2$ bills, i.e. that x'_R saves $\text{sav}_{x'_R}^\square = 0$ bills. We are done if w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins. From here on we assume that $\text{rcv}_{w_R}^\circ = 1$ holds, which is the minimum number of coins by Lemma 46. If x_L saves $\text{sav}_{x_L}^\circ = 1$ coin, then we are done again. So assume that we have $\text{sav}_{x_L}^\circ = 0$, i.e. that $\text{pay}_{x_L}^\circ = \text{PAY}_{x_L}^\circ = \Delta - 2$ transfers leave x_L , say to endpoints $w_R^1, \dots, w_R^{\Delta-2}$, see Figure 23.

We show a contradiction. After x'_R and x_L become matched, i.e. in step $s + 1$, each endpoint w_R^i has degree at most $d_{s+1}(w_R^i) \leq 1$ by definition, and endpoint w_R has degree at most $d_{s+1}(w_R) \leq 1$ due to Lemma 46, since we assumed that $\text{rcv}_{w_R}^\circ = 1$ holds.

First, we claim that in step $s + 1$ at least one of w_R and the w_R^i is not isolated and has degree exactly one. Why? Since (x'_R, y_L) is a large donation, in step s when x'_R and x_L

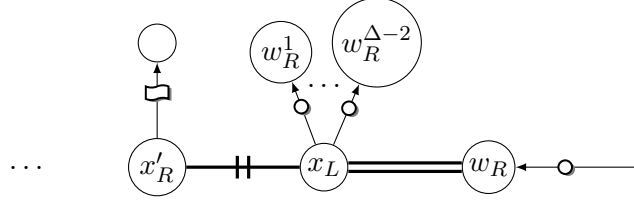


Figure 23: Node x'_R pays a large donation, node x_L pays $\Delta - 2$ transfers, endpoint w_R receives one transfer (not all edges are drawn)

become matched partition R contains exactly $\Delta - 2$ degree-1 endpoints by Definition 41. Since degrees drop by at most one in step s , at most $\Delta - 2$ endpoints become isolated, which holds in particular for w_R and the w_R^i . Therefore at least one of w_R and the w_R^i is not isolated in step $s + 1$ and consequently has degree *exactly* one.

Now observe that since (x'_R, y_L) is a large donation, in step s node x'_R is the only degree-1 node in R which is not an endpoint. Since x'_R becomes matched in step s , in step $s + 1$ all degree-1 nodes in R are endpoints. This contradicts Lemma 42. \square

4.3 All Components are Balanced

To complete the proof of Theorem 35 we proceed as follows. First we formally define the sets I_L and I_R of left resp. right increase nodes of a path. We may only add revenues and savings of both sets, if one does not contain a node of the other: we show that both sets are disjoint after the definition. Thereafter in Lemma 50 we complete the proof that paths are balanced. Finally we show that singletons are balanced as well.

Definition 48 (Left Increase Nodes). *Let X be a path created in step t . The set I_L of left increase nodes for X contains the following nodes, cf. Table 2.*

- We have $\{u_L, w_R\} \subseteq I_L$ for node u_L matched to create X and its “nearest” endpoint w_R .
- If X is not created in the step for w_R , then I_L additionally contains the nodes x'_R and x_L matched in the step for w_R .
- If in step $t + 1$ partition R contains an endpoint w with degree $d_{t+1}(w) = 1$, then I_L additionally contains node v_R (whether or not v_R is a node of X).

The set I_R of right increase nodes is defined symmetrically, i.e. by swapping L and R .

Lemma 49. *We have $I_L \cap I_R = \emptyset$.*

Proof. We discuss the left increase nodes w_R, v_R, x'_R and u_L, x_L in turn and show for each one: if it is contained in I_L then it is not contained in I_R .

Table 2: The set I_L of left increase nodes for path X

		Partition R contains a degree-1 endpoint after creation of X ?	
		no	yes
Path X created in step for endpoint w_R ?	yes ($\{u_L, u_R\} = \{x_L, x'_R\}$)	u_L, w_R	u_L, w_R, v_R
	no ($\{u_L, u_R\} \neq \{x_L, x'_R\}$)	u_L, w_R, x_L, x'_R	u_L, w_R, x_L, x'_R, v_R

Endpoint w_R is not a right increase node, since $w_R \neq w_L$ holds and all other nodes in I_R are M -matched whereas w_R is not.

To show that v_R is not a right increase node, we have to compare with all M -matched nodes in $I_R \cap R$, namely u_R and x_R : node v_R becomes matched with degree one whereas both u_R and x_R become matched when being incident with an M -edge and an M^* -edge.

Next, we show that x'_R is not a right increase node. As for v_R we only have to compare with u_R and x_R . If $\{u_L, u_R\} \neq \{x_L, x'_R\}$ holds, then observe that we have $x'_R \neq u_R$ as well as $x'_R \neq x_R$, since x_R becomes matched in the step for endpoint $w_L \neq w_R$. If, however, we have $\{u_L, u_R\} = \{x_L, x'_R\}$, then x'_R is not a left increase node in the first place.

To show that u_L and x_L are not right increase nodes, we have to compare with nodes x'_L and v_L (all other nodes are not M -matched or belong to R). The result now follows by applying the previous arguments for x'_R and v_R analogously to x'_L and v_L . \square

Completing the Proof. We are now ready to show that all paths and singletons are balanced.

Lemma 50. *Bound (6 Path_{K_S}) holds, i.e. a path X has balance at least*

$$\text{bal}_X \geq 2\Delta - \text{PAY}_X = 2\Delta - m_X 2(\Delta - 2) = \text{BAL}_X.$$

In particular, revenues and savings of nodes in I_L and I_R add up to at least 2Δ .

Proof. By (14), payments of X are bounded by at most $\text{PAY}_X = m_X 2(\Delta - 2)$ coins and bills. Consequently, revenues and savings of at least 2Δ imply that $\text{bal}_X \geq \text{BAL}_X$ holds.

We show that revenues and savings of nodes in I_L add up to at least Δ . The argument for I_R is symmetric. Since we have $I_L \cap I_R = \emptyset$ by Lemma 49, total revenues and savings add up to at least 2Δ .

Assume that KARPSIPSER creates X in step t . We distinguish cases based on whether partition R contains a degree-1 endpoint in step $t + 1$.

No Degree-1 Endpoint. First, we argue that node $u_L \in I_L$, matched in step t , pays $\text{pay}_{u_L}^\circ = 0$ coins. Why? No endpoint becomes isolated in step t , since in the reduced graph G_t each endpoint w has degree at least $d_t(w) \geq 2$ (as have u_L and u_R) and in G_{t+1} endpoint w has degree at least $d_{t+1}(w) \geq 1$ (since degrees drop by at most one). Since we assume that no degree-1 endpoint exists in step $t + 1$, each endpoint has degree at least two. Hence by Definition 10 (**Transfer**), no transfer leaves u_L .

Also, no donation leaves u_L by definition, i.e. we have $\text{pay}_{u_L}^\square = 0$. Consequently, by Definition 44 the savings of u_L are

$$\text{sav}_{u_L} = \Delta - 2.$$

Recall that in step $t + 1$ all endpoints have degree at least two, in particular the endpoint w_R of X . We distinguish whether step t is the step for w_R , i.e. whether $\{u_L, u_R\} = \{x_L, x'_R\}$ holds. If so, then in step $t + 1$ the M^* -edge $\{u_L, w_R\}$ of w_R is removed from the graph and w_R is incident with two F -edges. Therefore w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins by Lemma 46 and revenues and savings of I_L -nodes sum up to $\text{sav}_{u_L} + \text{rcv}_{w_R}^\circ = \Delta$. Otherwise the step for w_R happens after creation of X and we have $\text{rcv}_{w_R}^\circ + \text{sav}_{x_L} + \text{sav}_{x'_R} \geq 2$ by Lemma 47: again, the total “increase” is $\text{sav}_{u_L} + \text{rcv}_{w_R}^\circ + \text{sav}_{x_L} + \text{sav}_{x'_R} \geq \Delta$.

There is a Degree-1 Endpoint. Recall that in this case we have $v_R \in I_L$. We distinguish the following four cases for node v_R , which are restated below before their respective analysis.

- 1.–3. First, assume that X is created before the step for w_R , i.e. that $\{u_L, u_R\} \neq \{x_L, x'_R\}$ holds. We distinguish three cases for v_R .

For cases 1. and 2. assume that v_R is a node in X . We distinguish whether v_R becomes matched in the step for w_R , i.e. if we have $v_R \neq x'_R$ or $v_R = x'_R$.

Node v_R has degree $d_{t+1}(v_R) = 1$ after creation of X and we have $\{u_L, v_R\} \in E$, hence we get $\{v_R, u_L\} \in M^*$, since all other M -matched R -nodes of X have degree at least two in step $t + 1$.

In case 3. node v_R belongs to a component $Y \neq X$.

4. Here we assume that $\{u_L, u_R\} = \{x_L, x'_R\}$ holds.

In this case node v_R belongs to a component $Y \neq X$. Why? First, observe that there is only one M -matched node of X which could have degree one after creation of X , namely the M^* -neighbor of x'_R , which belongs to *partition L*. However, node v_R is defined to be the first degree-1 node becoming M -matched after creation of X in *partition R*.

Observe that only in cases 3. and 4. node v_R does not belong to X and we have to consider v_R paying a donation.

1. $\{u_L, u_R\} \neq \{x_L, x'_R\}$, v_R belongs to X , $v_R \neq x'_R$. Node v_R pays no transfer or donation, since by definition v_R has degree one when it becomes matched and since v_R belongs to the same path as u_L . We get $\text{pay}_{v_R}^\circ = \text{pay}_{v_R}^\square = 0$ and hence $\text{sav}_{v_R} = \Delta - 2$. By assumption, the step for endpoint w_R happens after creation of X . Moreover, since we have $v_R \neq x'_R$ node v_R does not become matched in the step for w_R , see Figure 24. Therefore we may sum savings of v_R and revenues and savings of nodes x'_R , x_L , and w_R , which are at least $\text{rcv}_{w_R}^\circ + \text{sav}_{x_L} + \text{sav}_{x'_R} \geq 2$ by Lemma 47. In total, savings and revenues add up to at least Δ .

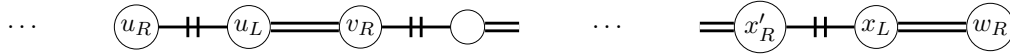


Figure 24: The path X in case 1 (not all edges are drawn)

2. $\{u_L, u_R\} \neq \{x_L, x'_R\}$, v_R belongs to X , $v_R = x'_R$. Recall that $\{u_L, v_R\}$ is an M^* -edge of X , since v_R is a neighbor of u_L and belongs to X , cf. Figure 25. As in the previous case, the savings of v_R are $\text{sav}_{v_R} = \Delta - 2$. So we are done with a total of Δ if w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins.

From here on, assume that w_R receives $\text{rcv}_{w_R}^\circ = 1$ coin, which is minimum by Lemma 46. It suffices to show that u_L or x_L saves a coin, since then we have $\text{sav}_{v_R} + \text{rcv}_{w_R}^\circ + \text{sav}_{u_L}^\circ + \text{sav}_{x_L}^\circ \geq \Delta$.

Assume that both u_L and x_L save $\text{sav}_{u_L}^\circ = \text{sav}_{x_L}^\circ = 0$ coins. Then both nodes pay $\text{pay}_{u_L}^\circ = \text{pay}_{x_L}^\circ = \Delta - 2$ coins, see Figure 25. We show a contradiction to Lemma 42. Therefore we argue that, after v_R and x_L become matched, there is a degree-1 node in R and all degree-1 nodes in R are path endpoints.

In step $t + 1$, destination endpoints of transfers from u_L have degree at most one, by definition; moreover, these endpoint have degree *exactly* one in step $t + 1$, since all degrees are at least two in step t (when X is being created) and in a bipartite graph degrees drop

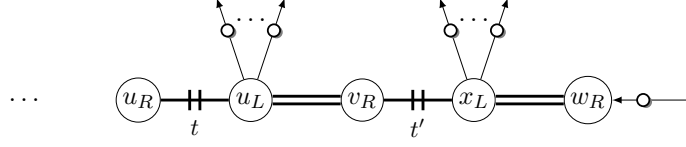


Figure 25: The path X in case 2 (not all edges are drawn)

by at most one. We denote the step when v_R becomes matched as step t' , for $t' \geq t + 1$. First, we argue that throughout steps $t + 1, \dots, t'$ the set of degree-1 nodes in R does not change. Why? Since v_R has degree $d_{t+1}(v_R) = \dots = d_{t'}(v_R) = 1$ and v_R is the first node in R becoming matched after step t (by definition), in each of steps $t + 1, \dots, t' - 1$ a degree-1 node *in partition L* becomes matched with an R -node of degree larger than one.

In steps $t + 1$ and $t' + 1$, destination endpoints of transfers from u_L resp. x_L have degree at most one, by definition. Node w_R has degree at most $d_{t'+1}(w_R) \leq 1$ after the step for w_R as well, by Lemma 46, since we have assumed that w_R receives $\text{rcv}_{w_R}^\circ = 1$ coin. Observe that the number of endpoints neighboring u_L (in G) is $\Delta - 2$, while x_L has $\Delta - 1$ neighboring endpoints (in G), namely w_R and the destination endpoints of transfers from x_L .

To obtain the desired contradiction, we study how many endpoints are neighbors of both u_L and x_L . Recall that all endpoints neighboring u_L and x_L have degree at least 2 in the creation step t of X , and that their degrees drop by at most one in each of steps t and t' (the set of degree-1 endpoints does not change in between these two steps, as shown above). Consequently, after $\{v_R, x_L\}$ is picked, i.e. in step $t' + 1$, we have the following. For at most $\Delta - 2$ of the endpoints neighboring u_L and x_L the degree is reduced to zero, since u_L is incident with at most $\Delta - 2$ endpoints. Therefore there is at least one endpoint neighboring x_L (in G) which still has degree exactly one. Moreover, in step $t' + 1$ all degree-1 nodes in partition R are endpoints, since both u_L and x_L have a “maximum” number of neighboring endpoints (in G) and both u_R and v_R are now matched. By Lemma 42 we obtain the desired contradiction.

3. $\{u_L, u_R\} \neq \{x_L, x'_R\}$, v_R belongs to $Y \neq X$. We have $\text{rcv}_{w_R}^\circ + \text{sav}_{x_L} + \text{sav}_{x'_R} \geq 2$ by Lemma 47, since X is not created in the step for w_R . Therefore it remains to prove additional savings and revenues of at least $\Delta - 2$. If u_L pays $\text{pay}_{u_L}^\circ = 0$ coins, then u_L saves $\text{sav}_{u_L} = \text{PAY}_{u_L} = \Delta - 2$ coins and bills by Definition 44, since u_L also pays $\text{pay}_{u_L}^\square = 0$ bills by Definition 26 (Donation). So assume that u_L pays $\text{pay}_{u_L}^\circ > 0$ coins. Then we have $\text{rcv}_{u_L}^\square + \text{sav}_{u_L}^\circ \geq \Delta - 2$ by Lemma 45. In both cases, revenues and savings of nodes u_L, x'_R, x_L , and w_R sum up to at least Δ .

4. $\{u_L, u_R\} = \{x_L, x'_R\}$ (v_R belongs to $Y \neq X$). Since both u_L and v_R are M -matched, the F -edge $\{u_L, v_R\}$ is not a transfer and u_L saves at least $\text{sav}_{u_L}^\circ \geq 1$ coin. In particular, we have $\text{sav}_{u_L} \geq 1$, since u_L pays no donation by definition.

Recall that each endpoint has degree at least two in the creation step t of path X , when u_L becomes matched. Therefore a destination endpoint of a transfer from u_L has degree exactly $d_{t+1}(w)=1$ after creation of X , since degrees drop by at most one in bipartite graphs, cf. Figure 26. Endpoint w_R has degree $d_{t+1}(w_R) = 1$ if and only if w_R receives exactly $\text{rcv}_{w_R}^\circ = 1$ coin, as a consequence of Lemma 46.

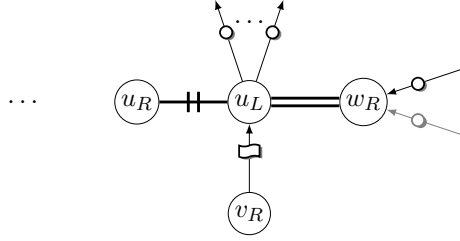


Figure 26: The path X in case 4, the gray transfer to w_R is optional (not all edges are drawn)

Consider the cases that w_R receives $\text{rcv}_{w_R}^\circ = 2$ coins or that u_L saves $\text{sav}_{u_L}^\circ \geq 2$ coins. In either case revenues and savings of w_R and u_L sum up to at least $\text{rcv}_{w_R}^\circ + \text{sav}_{u_L}^\circ \geq 3$, since we have $\text{rcv}_{w_R}^\circ \geq 1$ by Lemma 46 and $\text{sav}_{u_L}^\circ \geq 1$ as shown above. To push up savings and revenues to at least Δ , we show that there is a small donation (v_R, u_L) in which we move $\text{pay}_{v_R}^\square = \text{rcv}_{u_L}^\square = \Delta - 3$ bills to X . Therefore we have to argue that in step $t + 1$ there are at least one and at most $\Delta - 3$ degree-1 endpoints in partition R , cf. Definition 41. There is at least one degree-1 endpoint, by assumption. There are at most $\Delta - 3$ endpoints, for the following reasons. First, if we have $\text{rcv}_{w_R}^\circ = 2$ then w_R has degree at least $d_{t+1}(w_R) \geq 2$ by Lemma 46, i.e. neighbors u_R, v_R , and w_R of u_L are not degree-1 endpoints. Secondly, if we have $\text{sav}_{u_L}^\circ \geq 2$ then at most $\Delta - 4$ endpoints receive a transfer from u_L and have degree exactly one in step $t + 1$: endpoint w_R may also have degree one without increasing the number of degree-1 endpoints too far.

Now consider the case that w_R receives exactly $\text{rcv}_{w_R}^\circ = 1$ coin and u_L saves only $\text{sav}_{u_L}^\circ = 1$ coin. We prove additional revenues of $\Delta - 2$. Node u_L pays $\text{pay}_{u_L}^\circ = \Delta - 3$ coins, say to endpoints $w_R^1, \dots, w_R^{\Delta-3}$. Observe that in step $t + 1$ partition R contains exactly $\Delta - 2$ degree-1 endpoints, namely w_R and the w_R^i , and that v_R is the only degree-1 node in R which is not an endpoint. Therefore, by Definition 41, in a large

donation (v_R, u_L) we move additional $\text{rcv}_{u_L}^\square = \Delta - 2$ bills to X , which push total savings and revenues up to at least Δ . \square

Lemma 51. *Bound (5 Singleton) holds, i.e. a singleton Z has balance at least $\text{bal}_Z \geq \text{BAL}_Z = -2(\Delta - 2)$.*

Proof. Singleton Z receives $\text{rcv}_Z = \text{rcv}_Z^\circ = \text{rcv}_Z^\square = 0$ coins and bills, since only path nodes receive funds. The balance is defined as $\text{bal}_Z = \text{rcv}_Z - \text{pay}_Z = -\text{pay}_Z$, hence it suffices to show that Z pays at most $\text{pay}_Z \leq 2(\Delta - 2)$ coins and bills.

Recall that an M -matched node pays transfers or donations, but not both, and that at most one donation leaves each node. Let z_L and z_R denote the nodes of Z . We distinguish the following four cases: at most one of z_L or z_R pays funds, both z_L and z_R pay a donation, both pay a transfer, or w.l.o.g. (z_L, u_R) is a donation and z_R pays a transfer.

At Most One Node Pays Funds. We are done if none of z_L and z_R pays funds. So assume that z_L pays funds and z_R pays $\text{pay}_{z_R}^\circ = \text{pay}_{z_R}^\square = 0$ coins and bills. Node z_L pays at most $\text{pay}_{z_L}^\circ \leq \text{PAY}_{z_L}^\circ = \Delta - 1$ coins by Lemma 12 a), at most $\text{pay}_{z_L}^\square \leq \text{PAY}_{z_L}^\square = \Delta - 2$ bills by Definition 41, and z_L does not pay both coins *and* bills. We obtain $\text{pay}_{z_L} \leq \Delta - 1 \leq 2(\Delta - 2)$ and are done, since $\Delta \geq 3$ holds.

Both Nodes Pay a Donation. Exactly two donations leave Z and Z pays no coins. By Definition 41, in each donation we move at most $\Delta - 2$ bills. We get $\text{pay}_Z^\square \leq 2(\Delta - 2)$.

Both Nodes Pay a Transfer. It suffices to show that each of z_L and z_R pays at most $\Delta - 2$ coins, since both pay no donation and hence $\text{pay}_{z_L}^\square = \text{pay}_{z_R}^\square = 0$ bills.

Assume that z_L pays more coins, namely the maximum of $\text{pay}_{z_L}^\circ = \Delta - 1$ coins, cf. Lemma 12 a). We show a contradiction. When z_L and z_R become matched, say in step t , then node z_L has degree at least $d_t(z_L) \geq \Delta \geq 3$, since z_L is incident with $\Delta - 1$ transfers and with edge $\{z_L, z_R\}$, see Figure 27. By assumption, each of z_L and z_R pays a transfer, therefore both z_L and z_R are incident with an F -edge in step t and by definition of the KARPSIPSER algorithm all nodes in G_t have degree at least two. Since degrees drop by at most one in each step, after z_L and z_R become matched a destination endpoint w of a transfer from z_L has degree exactly $d_{t+1}(w) = 1$. Furthermore, the $\Delta - 1$ destination endpoints of transfers from z_L are the only degree-1 nodes in partition R . A contradiction to Lemma 42.

The analogous argument applies to z_R , i.e. we have $\text{pay}_{z_R}^\circ \leq \Delta - 2$. In total, we get $\text{pay}_Z = \text{pay}_{\{z_L, z_R\}}^\circ \leq 2(\Delta - 2)$.

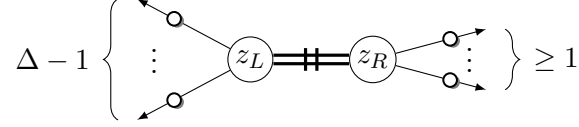


Figure 27: Node z_L pays $\Delta - 1$ transfers (not all edges are drawn)

A Donation (z_L, u_R) Leaves z_L and z_R Pays a Transfer. We are done if (z_L, u_R) is a small donation in which we move $\text{pay}_{z_L}^\square = \Delta - 3$ bills. Why? Since z_L pays $\text{pay}_{z_L}^\circ = 0$ coins and z_R pays at most $\text{pay}_{z_R}^\circ \leq \text{PAY}_{z_R}^\circ = \Delta - 1$ coins and $\text{pay}_{z_R}^\square = 0$ bills: in total, we have $\text{pay}_{\{z_L, z_R\}} \leq \Delta - 3 + \Delta - 1 = 2(\Delta - 2)$.

Assume that (z_L, u_R) is a large donation in which we move $\text{pay}_{z_L}^\square = \Delta - 2$ bills. It suffices to bound the number of coins paid by z_R to at most $\text{pay}_{z_R}^\circ \leq \Delta - 2$. (Note that the argument of the previous case cannot be applied, since it depends on the assumption that z_L becomes matched with degree Δ . Here, however, node z_L becomes matched with degree one, since it pays a donation.)

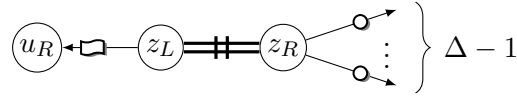


Figure 28: Node z_L pays $\Delta - 2$ bills in a large donation (not all edges are drawn)

Assuming that z_R pays $\text{pay}_{z_R}^\circ = \Delta - 1$ coins, say to endpoints $w_L^1, \dots, w_L^{\Delta-1}$, we show a contradiction, cf. Figure 28. Consider the step t when z_L becomes matched. By definition of (z_L, u_R) , partition L contains exactly $\Delta - 2$ degree-1 path endpoints and no further degree-1 nodes but z_L . In particular, at least one of the $\Delta - 1$ many w_L^i has degree at least $d_t(w_L^i) \geq 2$. Hence after z_L and z_R become matched, i.e. in step $t + 1$, at least one of the w_L^i has degree exactly $d_{t+1}(w_L^i) = 1$, since the degree of w_L^i drops from at least $d_t(w_L^i) \geq 2$ to at most $d_{t+1}(w_L^i) \leq 1$ (by definition) and by exactly one (since the graph is bipartite). Furthermore, since z_L is now matched, all degree-1 nodes in L are path endpoints. This contradicts Lemma 42. \square

5 Approximation Bounds for Greedy Matching

Throughout this section we systematically study limitations to the approximation performance of greedy algorithms for matching. To obtain inapproximability results we employ the framework of *adaptive priority algorithms*, introduced by Borodin, Nielsen, and Rackoff [BNR03]. This framework enables us to define large classes of “greedy-like” algorithms. For the obtained classes, the framework also provides methods for obtaining bounds on their approximation performance.

In particular, our definitions allow us to obtain tight inapproximability bounds which match the performance guarantees $\frac{\Delta}{2\Delta-2}$ resp. $\frac{\Delta-1}{2\Delta-3}$ proven for the KARPSSIPER algorithm and 1-2-GREEDY on graphs of degree at most Δ , cf. Section 3 resp. Section 4.

Moreover, we emphasize that KARPSSIPER and 1-2-GREEDY are an intriguing combination of both simplicity and approximation performance. Therefore we analyze limitations of the following types of algorithms.

- Algorithms which are unable to incorporate degree information into the choice of the next edge.

Such algorithms cannot beat factor $\frac{1}{2}$, even for most simple graphs.

- Algorithms which are able to utilize degree information of both nodes in the next edge, which may pick several edges at a time, or which are even capable of exploiting neighborhood relationships between nodes.

Such algorithm achieve approximation ratio only $\frac{1}{2}$ for large graphs.

Organization of this Section. In Section 5.1 we present and discuss the adaptive priority algorithm framework. The definitions of our classes of algorithms—and the motivation for the respective definition—are given in Section 5.2. In Section 5.3 we first present an overview of our inapproximability bounds and then show the constructions used to obtain these bounds.

5.1 The Priority Algorithm Framework

To study limitations of greedy algorithms, Borodin, Nielsen and Rackoff [BNR03] introduced the framework of *adaptive priority algorithms*.

An adaptive priority algorithm A is defined relative to the notion of a *data item*. The definition of a data item depends on the problem at hand, and each data item represents part of the input (e.g. an edge in a graph). An input instance I is modeled as a set of data items.

When A starts its computations, instance I is unknown to A . Algorithm A computes in *rounds*, where in each round algorithm A processes a new data item $i \in I$ in the input. For each data item i algorithm A has to commit to an *irrevocable decision* for part of its solution based on an *irrevocable decision* for i (e.g. whether to pick an edge or not).

Algorithm A follows the scheme given in Algorithm 3, cf. [BBLM10]. Before computations start, algorithm A is given *a priori knowledge* K on the input instance I (e.g. the number of nodes in the input graph). However, the data items in I remain *unseen*, as indicated by renaming I to U in Algorithm 3.

In a given round, how does A acquire the next data item i ? Algorithm A determines a total *priority order* π of *all possible* data items, see Line 5; then A *receives* an unseen data item $i \in U$ of highest priority in π (all data items in $U \setminus \{i\}$ remain unseen). In particular, the function f which computes π has *no* access to I or U . However, function f may access the given a priori knowledge K . Moreover, function f may access all knowledge gathered in previous rounds, which is encoded in the history of previous decisions H . (We discuss the properties of function f in more detail in Section 5.1.) Since algorithm A may incorporate all knowledge gathered in earlier rounds into a “request” for the next data item, algorithm A reacts *adaptively* to previous computations.

For the current data item i , algorithm A has to make an irrevocable decision in Line 7 (e.g. whether to include an edge in a matching or not). This decision is made by a function g , which has to respect all constraints on legal solutions for the problem at hand (e.g. in a matching no node may be matched more than once). As for f , function g may consider a priori knowledge and all gathered knowledge. (We discuss the properties of function g in more detail in Section 5.1.)

Data item i and the according decision d are remembered by appending (i, d) to the history H of computations: this way algorithm A gathers knowledge about I .

After data item i has been inspected and a decision for i has been made, data item i is removed from the set of unseen data items U , see Line 9.

As soon as no more unseen data items are left, algorithm A returns the solution, which is fully encoded in the history H of received data items and their decisions.

The notion of being “greedy” is captured by two properties of priority algorithms, namely by irrevocable decisions as well as by allowing to submit priority orders. In particular, the latter property distinguishes greedy algorithms from online algorithms: online algorithms have no control over which data item is received next.

Algorithm 3 Scheme of an adaptive priority algorithm A on input I .

1. $K \leftarrow$ a priori knowledge on I
 2. $U \leftarrow I$ ▷ data items in I are *unseen*²³
 3. $H \leftarrow$ empty list ▷ “history” of data items and their *irrevocable decisions*
 4. **while** $U \neq \emptyset$ **do** ▷ proceed with next *round*²³
 5. $\pi \leftarrow f(H, K)$ ▷ determine *total* priority order
 6. $i \leftarrow \min_{\pi} U$ ▷ the highest priority unseen data item²³
 7. $d \leftarrow g(i, \pi, H, K)$ ▷ make an irrevocable decision d for i
 8. $H \leftarrow H \cdot (i, d)$ ▷ remember data item and decision
 9. $U \leftarrow U \setminus (\{i\})$ ▷ remove now non-eligible data items²³
 10. **end while**
 11. **return** H ▷ solution is fully encoded in H
-

Introductory Examples. The framework of adaptive priority algorithms was adapted to graph problems by Davis and Impagliazzo [DI09] and Borodin, Boyar, Larsen and Mirmohammadi [BBLM10]. To tailor the scheme given in Algorithm 3 to a particular graph problem, one has to specify a type of data items and possible irrevocable decisions. A particular priority algorithm for the given graph problem is then obtained by implementing functions f and g .

First, we briefly bring to mind Kruskal’s algorithm for the Minimum Spanning Tree Problem. Given an undirected weighted input graph G , this algorithm first sorts all edges of G by weight in ascending order. All structural information about G obtained in this preprocessing phase is ignored throughout the rest of the algorithm. Then, Kruskal’s algorithm repeatedly picks the cheapest edge which does not close a cycle in the forest of previously picked edges.

To implement Kruskal’s algorithm as an adaptive priority algorithm, the graph G can e.g. be modeled as a set of weighted edges $\langle u, v; w \rangle$, where u and v are neighbors in G and w is the weight of edge $\{u, v\}$. The possible decisions are to include the edge of a data item in the spanning tree or not. Function f computes a total order $\dots, \langle u_1, v_1; w_1 \rangle, \langle u_2, v_2; w_2 \rangle, \dots$ on *all possible* data items such that $w_i \leq w_j \Leftrightarrow i < j$ holds. Observe that the actual set of edges in G does not need to be known to compute π . The decision function g tests whether the nodes in the current edge belong to the same tree or not (note that the forest can be constructed from H at any time). Based on the outcome of g , the current edge is included in the spanning tree or not.

²³The algorithm may not look at data items in I or U .

Fixed Priority Algorithms. The function f in the implementation of Kruskal’s algorithm as a priority algorithm does not depend on previously gathered knowledge, since it outputs the same priority order in each round. This type of priority algorithm is called a *fixed* priority algorithm, since it does not adaptively use its current knowledge.

In contrast, Prim’s minimum spanning tree algorithm chooses a cheapest possible edge out of those edges which connect a new node to the previously computed “sub-spanning tree”. Therefore, function f *adaptively* depends on the outcome of prior computations when determining priorities for possible next edges. Hence Prim’s algorithm cannot be implemented as a fixed priority algorithm.

Fixed priority algorithms are a subclass of adaptive priority algorithms. Generally, the power of fixed and adaptive priority algorithms is not the same. Whereas Dijkstra’s shortest paths algorithm can be implemented as an adaptive priority algorithm and computes exact results, fixed priority algorithms cannot even give approximate solutions [DI09].

In this thesis we focus on adaptive priority algorithms only. Why? Since algorithms such as MINGREEDY or KARSIPSER cannot be implemented as fixed priority algorithms. Intuitively, in each step node degrees change depending on which edge is picked, therefore the next edge to be picked cannot be determined in advance.

Approximation Algorithms. Kruskal’s algorithm, Prim’s algorithm, and Dijkstra’s algorithm compute optimal solutions. Approximate greedy algorithms can be implemented as priority algorithms as well. Consider e.g. the nearest neighbor heuristic for the metric traveling salesman problem: starting with an arbitrary city s , a tour is constructed by repeatedly visiting an unvisited city closest to the current city, and eventually returning to s . This heuristic has approximation ratio $\Theta(\log_2 n)$ on n -node instances [RSL77]. It can be implemented as an adaptive priority algorithm as follows. We use data items of the form $\langle i; d_1, \dots, d_{i-1}, 0, d_{i+1}, \dots, d_n \rangle$ where i is a city and the d_j are the distances from i to the other cities. A nearest neighbor of the current city c has highest priority in the total order on all possible data items sorted ascending by d_c .

Discussion. Adaptive priority algorithms are not constrained in the use of time or space for their computations of priority orders and decisions. The only restriction on functions f and g is that, in each step, future data items may not be considered in the computation (in fact, these function may be non-computable, see [DI09] for a discussion, and f may compute non-finite orders, e.g. for real-valued data items like weighted edges).

Hence, the only limitation of an adaptive priority algorithm is the requirement to unveil an input instance piece by piece, i.e. one data item per round, and to construct a part of the solution once and forever for each new data item.

Since adaptive priority algorithms are limited solely by this property of “greedily” constructing a solution, inapproximability results apply to correspondingly large classes of algorithms.

Data Items. For any given optimization problem, the definition of data items (e.g. a data item might specify an edge in a graph, it may additionally state the degrees of both nodes, or it may contain the entire set of neighbors of both nodes) has strong impact on the range of algorithms captured in the obtained class.

Therefore, data items have to be chosen carefully for the following reasons. Assume that the definition of data items allows an algorithm to acquire a large amount of knowledge about the input (per round). The obtained class contains algorithms whose computations of priority orders and decisions process this vast knowledge in sophisticated ways: such algorithms do not comply with the common understanding of a simple greedy heuristic. Moreover, giving such “absurd” power to an algorithm implies that inapproximability bounds are difficult to prove and that obtained bounds do not reflect limitations of reasonable algorithms but rather those of artificial algorithms.

Furthermore, if the problem at hand has polynomial time methods to compute exact solutions, then allowing powerful data items introduces the following additional complication. We can only obtain inapproximability bounds which are not informative: they are meaningful only for algorithms with larger resource demands than those of known efficient algorithms with approximation factor 1.

Applications from the Literature. The model of adaptive priority algorithms was successfully applied to obtain inapproximability bounds for e.g. Scheduling [BNR03], Facility Location and Set Cover [AB04], Max-Sat [Pol11], Sum-Coloring [BIYZ12], graph problems like Steiner-Tree or Independent-Set [DI09], or randomized matching in general graphs [Pol12, BP15].

5.1.1 Obtaining Inapproximability Results

Let a formulation of an optimization problem P in terms of data items and possible irrevocable decisions be given. If functions f and g are left unspecified, then we obtain an entire class A_P of adaptive priority algorithms for P . Inapproximability results are obtained similar to the adversarial arguments found in the analysis of the competitive ratio of online algorithms. Let A be an algorithm in A_P . In the *adaptive priority game*,

algorithm A plays against an adversary B . The adversary processes the submitted orders and the irrevocable decisions of A to construct a hard input instance for A . In particular, adversary B takes control over Lines 2 and 6 in Algorithm 3 (marked with bold line numbers):

Line 2: Observe that before the game starts adversary B is free to construct any instance which is consistent with the given a priori knowledge: if e.g. the number n of nodes in the input graph is announced to A before the first round, then B is free to construct *any* graph with n nodes. This gives B freedom to react to the various moves of A . We denote the set of possible instances as \mathcal{I} and note that $|\mathcal{I}| \geq 1$ holds in general.

Line 6: This is where adversary B pursues its plan to construct a hard instance. The goal of adversary B is to present a data item i which only allows poor decisions by A .

Consistency. Of course, adversary B has to act *consistently*. Consider the priority order π of the current round. Since the presented data item i has to have highest priority in π (among unseen data items), adversary B may not present a data item of higher priority than i (in π) in a later round.

Intuitively, if B acts consistently then the following holds: giving the final instance as input to A —without an adversary being involved—produces the same sequence of submitted priority orders, received data items, and irrevocable decisions.

Therefore, the set of possible instances \mathcal{I} must be reduced to those instances I for which $i \in I$ holds and which are consistent with the previous game. In the end, set \mathcal{I} is reduced to one instance $\mathcal{I} = \{I\}$, namely an instance I for which the approximation performance of A is poor.

5.2 Adaptive Priority Algorithms for Matching

We only study deterministic algorithms. Whenever we discuss an algorithm whose definition depends on the use of randomization, like e.g. MINGREEDY, then we implicitly refer to the modified definition in which each random choice is replaced by a deterministic choice.

Throughout this section we present the hierarchy of classes of algorithms given in Figure 29. In the figure we indicate inclusion relations between classes and show to which classes the algorithms introduced in Section 1.1 belong. All our classes include the

class \mathcal{QC} of algorithms for the query commit problem, and therefore also the class \mathcal{VI} of vertex iterative algorithms.²⁴ Classes shaded with the same color are closely related and are therefore discussed together, as we outline next.

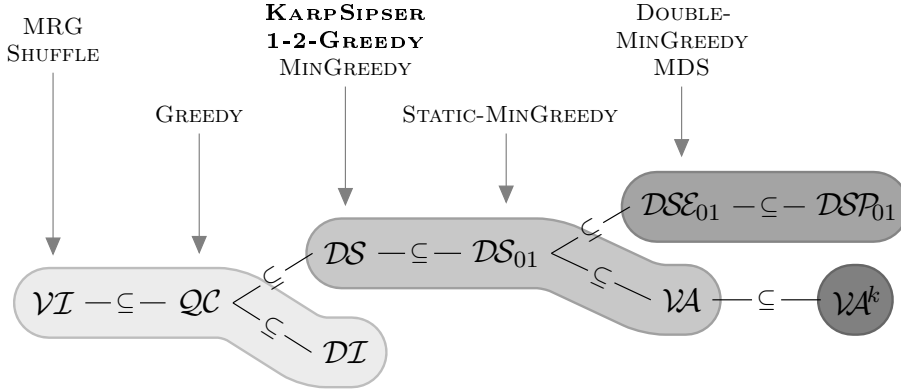


Figure 29: Inclusion relations between classes of algorithms (arrows point to smallest classes containing given algorithms, shades indicate closely related algorithms)

- In Section 5.2.1 we introduce the central class \mathcal{DS} in the hierarchy, which is called *degree sensitive* algorithms. Intuitively, each data item represents an edge in the graph and states the current degree of one of both nodes. Therefore algorithms KARPSSIPER and 1-2-GREEDY are contained in this class.

We build the definition of other classes upon the definition of \mathcal{DS} by varying the type of data items used (the rest of the definition of \mathcal{DS} remains unchanged). First, we introduce additional information on one node of an edge. Algorithms in class \mathcal{DS}_{01} may specify either the current degree of a node or its degree in the input graph. The entire neighborhood of one node may be specified by algorithms in class \mathcal{WA} . These algorithms have more fine-grained control than \mathcal{DS} -algorithms over which edge is picked next.

For bipartite graphs, we show that the KARPSSIPER algorithm has optimal performance among all \mathcal{DS}_{01} -algorithms. For general graphs, we show that 1-2-GREEDY has optimal performance even among all \mathcal{WA} -algorithms.

²⁴See Section 1.4.2 for a discussion of \mathcal{QC} and \mathcal{VI} as well as for the corresponding inclusion and membership relations indicated in the figure.

- In Section 5.2.2 we discuss data items providing degree information on both nodes of an edge. Algorithms in class \mathcal{DSE}_{01} contain “double-sided” algorithms such as the *minimum degree sum* algorithm MDS.

Such algorithms achieve an approximation ratio scarcely better than KARPSIPSER or 1-2-GREEDY.

\mathcal{DSP}_{01} is a class of algorithms with greater capabilities to construct a matching: in each step an entire set of edges belonging to one or several paths may be picked, instead of just one single edge. At first glance, a matching consisting only of long alternating paths constitutes a good approximation to a maximum matching. We will see, however, that \mathcal{DSP}_{01} -algorithms perform poorly on large graphs.

- In Section 5.2.3 we define the class \mathcal{DI} of *degree ignorant* algorithms which do not incorporate degree information into the choice of the next edge. Despite \mathcal{DI} -algorithms having much finer control over the next edge to be picked than \mathcal{QC} -algorithms (e.g. neighbor relationships may be specified), no stronger approximation ratio is achieved: algorithms in \mathcal{DI} fail with factor exactly $\frac{1}{2}$.

Consequently, using degree information is an indispensable requirement for a greedy matching algorithm to achieve a non-trivial approximation ratio.

- In Section 5.2.4 we study adaptive priority algorithms for the k -Hypergraph Matching Problem, which generalizes the standard matching problem. An algorithm in the class \mathcal{VA}^k achieves approximation factor exactly $\frac{1}{k}$, which is worst possible.

Therefore the success of greedy approaches for matching does not carry over to hypergraph matching.

5.2.1 Degree Sensitive Algorithms

First, in Definition 52 we present the central class \mathcal{DS} of *degree sensitive algorithms*, upon which we build the definitions of other classes. In Lemma 53 we show that \mathcal{DS} includes most of the popular greedy matching algorithms; in particular, \mathcal{DS} contains algorithms which depend on degree information such as MINGREEDY as well as algorithms like GREEDY which do not incorporate degree information into the choice of the next edge.

The class \mathcal{DS} was defined in [BW15]. Since \mathcal{DS} is the basis of further generalizations and several results in Section 5, we repeat its definition. Furthermore, we supplement the definition with basic properties of this class.

Definition 52 (Degree Sensitive Algorithm). *An adaptive priority algorithm A for the matching problem belongs to the class \mathcal{DS} of degree sensitive algorithms if the following holds:*

- a) *A priori, algorithm A is allowed to read the set of nodes in the input graph.*
- b) *Algorithm A computes on data items*

$$\langle u, d_u, v \rangle$$

which state two nodes u and v and an integer $d_u \geq 1$. If algorithm A receives data item $\langle u, d_u, v \rangle$ in round t , then node u has

- *current degree d_u and*
- *the neighbor v*

in the reduced graph G_t (the reduced graph does not contain edges incident with nodes matched before round t).

- c) *If A receives data item $\langle u, d_u, v \rangle$ then edge $\{u, v\}$ must be picked.*

The definition of \mathcal{DS} extends the so-called *edge model* of Davis and Impagliazzo [DI09] by introducing the degree of a node into each data item.

Modification of the Framework. The following two slight modifications of the original definition of an adaptive priority algorithm have to be taken care of.²⁵ (We denote by A an algorithm which adheres to the definition given in Algorithm 3, and by A' an algorithm in \mathcal{DS} .) Consider the update of the set U of unseen data items in Line 9.

1. When a data item received by A is removed from the set U of unseen data items, no other data item is removed. In particular, algorithm A receives *each* data item in the input instance.

Modification. In general, an algorithm $A' \in \mathcal{DS}$ does not receive every data item. Why? Whenever A' receives a data item $\langle u, d_u, v \rangle$, then edge $\{u, v\}$ is contained in the reduced graph (and must be added to the solution). Therefore edge $\{u, v\}$ is not incident with a node matched in a previous round. In particular, algorithm A' *does not* receive further data items representing edges incident with previously matched nodes.

We formalize this modification as follows. In the update of set U of unseen data items we do not only remove the current data item $\langle u, d_u, v \rangle$ but also each data item $\langle x, d_x, y \rangle$ with $\{x, y\} \cap \{u, v\} \neq \emptyset$.

²⁵These modifications require an according adjustment of the adaptive priority game. In particular, we have to tailor the definition of a *consistent* adversary. We do this in Section 5.3.1, before we prove our inapproximability bounds.

2. When a data item received by A is removed from the set U of unseen data items, the remaining data items are not changed.

Modification. For an algorithm $A' \in \mathcal{DS}$, however, certain remaining data items might change. Why? In the (non-reduced) input graph $G = G_1$, data item $\langle u, d_u, v \rangle \in G_1$ states the *initial* degree $d_u = d_1(u)$ of node u . However, in round $t > 1$ the data item $\langle u, d_u, v \rangle$ received by A states the *current* degree $d_u = d_t(u)$ of node u in the reduced graph G_t . In particular, we might have $d_t(u) < d_1(u)$ since the degree of u decreased during rounds $1, \dots, t-1$.

This modification is formalized as follows. Consider any round round $t \geq 1$. In the update of set U we replace each remaining unseen data item $\langle x, d_t(x), y \rangle$ with its updated version $\langle x, d_{t+1}(x), y \rangle$.

Lemma 53. *We have*

$$\begin{aligned} \text{KARPSIPSER} &\in \mathcal{DS}, \\ \text{1-2-GREEDY} &\in \mathcal{DS}, \\ \text{MINGREEDY} &\in \mathcal{DS}, && \text{as well as} \\ \mathcal{QC} &\subseteq \mathcal{DS}. \end{aligned}$$

Proof. We assume that the set of nodes in the input graph is $V = \{v_1, \dots, v_n\}$. Also, we use the following notation: a data item d with an asterisk $*$ in the middle position represents an (arbitrarily ordered) list of data items in which $*$ is replaced with all possible integers greater than zero, i.e. we have $\langle v_j, *, v_k \rangle = \langle v_j, d_1, v_k \rangle, \langle v_j, d_2, v_k \rangle, \dots$ where $\bigcup_{i \geq 1} \{d_i\} = \bigcup_{i \geq 1} \{i\}$ holds.

To implement the **KARPSIPSER** algorithm as a degree sensitive algorithm, in each round we first construct the temporary order

$$\langle v_{i_1}, *, v_{i_2} \rangle, \langle v_{i_1}, *, v_{i_3} \rangle, \dots, \langle v_{i_1}, *, v_{i_n} \rangle, \langle v_{i_2}, *, v_{i_3} \rangle, \dots, \langle v_{i_{n-1}}, *, v_{i_n} \rangle$$

for arbitrary i_1, \dots, i_n with $\bigcup_{j=1}^n \{i_j\} = \{1, \dots, n\}$. Each data item $\langle v_{i_j}, *, v_{i_{j'}} \rangle$ represents a possible edge, where we do not make a difference for the degree of node v_{i_j} . Therefore this order represents an arbitrary order of preferences for the choice of the next edge.

Now we move all data items $\langle v_{i_j}, 1, v_{i_{j'}} \rangle$ to the front of the order such that the relative order of the moved data items is unchanged.

Therefore, an arbitrary node of current degree 1 and its incident edge are preferred over any other edge, and if no degree-1 node exists an arbitrary edge is picked.

For **1-2-GREEDY** we proceed analogously: we move data items $\langle v_{i_j}, 1, v_{i_{j'}} \rangle$ to a “degree-1-block” in the front of the order and data items $\langle v_{i_j}, 2, v_{i_{j'}} \rangle$ between the initial degree-1 block and the remainder of the order.

Analogously, to implement **MINGREEDY** we build a “degree- i block” for each possible current degree i and order the resulting blocks ascending by degree.

Lastly, to verify the inclusion $\mathcal{QC} \subseteq \mathcal{DS}$ we use standard simulation arguments and show how to implement an algorithm $A \in \mathcal{QC}$ for the query commit problem as a degree sensitive algorithm $A' \in \mathcal{DS}$. This simulation has to obey the following property, since then A' computes the same solution as A . For every input graph, if the i -th edge picked by A is $\{u, v\}$, then in round i algorithm A' computes a priority order from which A' receives a data item such that nodes u and v are matched.

Recall that A probes for edges in the graph one by one and picks each edge for which the probe is *successful*, i.e. for which the edge is contained in the graph. One round of A' implements a series of unsuccessful probes and ends with an edge picked by A .

By e_i we denote the i -th edge picked by A (*not* the i -th probed edge). Assume that edges e_1, \dots, e_i were picked in the 1st, 2nd, \dots , i -th round of A' , respectively. In round $i+1$, algorithm A' has to construct the $i+1$ -th priority order, call it π_{i+1} , such that edge e_{i+1} is picked.

Algorithm A' initializes an empty order $\pi_{i+1} = \varepsilon$ which contains zero data items. In order to fill π_{i+1} , algorithm A' simulates A beginning with (but not including) picking edge e_i . Denote by $\{v_{j_1}, v_{k_1}\}$ the edge probed by A after picking edge e_i . Algorithm A' appends the order $\langle v_{j_1}, *, v_{k_1} \rangle$ to π_{i+1} .

- Assume that edge $\{v_{j_1}, v_{k_1}\}$ is in the graph. Since A picks this edge we have $e_{i+1} = \{v_{j_1}, v_{k_1}\}$. No matter how the rest of π_{i+1} is constructed, algorithm A' receives a data item $\langle v_{j_1}, d, v_{k_1} \rangle$, since a data item with the current degree d of v_{j_1} is contained in the sub-order $\langle v_{j_1}, *, v_{k_1} \rangle$ in the beginning of π_{i+1} . But when receiving data item $\langle v_{j_1}, d, v_{k_1} \rangle$, then algorithm A' has to pick edge $\{v_{j_1}, v_{k_1}\} = e_{i+1}$, i.e. the same edge as A . Thereafter, algorithm A' proceeds with round $i+2$, where a new priority order has to be constructed.
- Assume that the graph does not contain edge $\{v_{j_1}, v_{k_1}\}$. We have to show the construction of the remainder of π_{i+1} . Note that A does not pick edge $\{v_{j_1}, v_{k_1}\}$ and that A' does not receive a data item from $\langle v_{j_1}, *, v_{k_1} \rangle$. Therefore we can proceed recursively as follows. Assume that algorithm A probes for edge $\{v_{j_2}, v_{k_2}\}$ next. Then algorithm A' appends the order $\langle v_{j_2}, *, v_{k_2} \rangle$ to π_{i+1} . The argument that if the

graph contains edge $\{v_{j_2}, v_{k_2}\}$ algorithm A' receives a data item from $\langle v_{j_2}, *, v_{k_2} \rangle$ and picks the same edge $e_{i+1} = \{v_{j_2}, v_{k_2}\}$ as A is analogous as above.

To repeat the argument, assume that if A has probed for edges $\{v_{j_1}, v_{k_1}\}, \dots, \{v_{j_\ell}, v_{k_\ell}\}$ without success, then A probes for edge $\{v_{j_{\ell+1}}, v_{k_{\ell+1}}\}$ next. Algorithm A' constructs the priority order

$$\pi_{i+1} = \langle v_{j_1}, *, v_{k_1} \rangle, \langle v_{j_2}, *, v_{k_2} \rangle, \dots$$

We obtain the following. If edges $\{v_{j_1}, v_{k_1}\}, \dots, \{v_{j_\ell}, v_{k_\ell}\}$ are not contained in the graph but edge $\{v_{j_{\ell+1}}, v_{k_{\ell+1}}\}$ is, then A picks edge $e_{i+1} = \{v_{j_{\ell+1}}, v_{k_{\ell+1}}\}$ and A' receives a data item $\langle v_{j_{\ell+1}}, d, v_{k_{\ell+1}} \rangle$ and picks edge e_{i+1} as well.

Thereafter, algorithm A' proceeds with the next round $i + 2$, where a new priority order has to be constructed. \square

Note. W.r.t. the data items for the class \mathcal{DS} , algorithms GREEDY, MRG, and SHUFFLE also belong to the class of so called *fixed* priority algorithms. Such an algorithm may compute a priority order $\pi = d_1, d_2, \dots$ on data items d_i only once, namely before the first round. Only one “pass” over the order π is allowed, cf. [DI09]: in each round, only data items in π are regarded which have lower priority than all previously received data items.

Observe that GREEDY regards edges in an arbitrary order, MRG uses an arbitrary order on nodes and their incident edges, and SHUFFLE computes an “edge probe order” in form of a permutation of the nodes. All these orders on edges can be initialized before the first round. To transform an edge probe order into a priority order on data items, each edge $\{u, v\}$ is replaced with a sub-order containing a data item $\langle u, d_u, v \rangle$ for each possible degree d_u .

However, MINGREEDY is not a fixed priority algorithm: to give higher priority to nodes of small degree, data items containing small degrees have to be located in the beginning of a priority order. Similarly, 1-2-GREEDY and the KARPSSIPER algorithm are not fixed priority algorithms.

Generalizing Degree Sensitive Algorithms. To allow for more fine-grained edge selection routines, we generalize the definition of data items of degree sensitive algorithms. In particular, a data item

$$\langle u, d_u, v \rangle$$

is extended such that it exposes more information on node u than only its current degree. The rest of Definition 52 remains unchanged. In particular, for each received data item $\langle u, \dots, v \rangle$ the algorithm has to pick edge $\{u, v\}$.

- i. For the class \mathcal{DS}_{01} a data item

$$\langle u, d_u, b_u, v \rangle$$

contains an additional *bit* $b_u \in \{0, 1\}$.

Bit b_u determines the “reference” of degree d_u .

- If $b_u = 0$ holds, then d_u refers to the current degree of u in the reduced graph (as for \mathcal{DS}).
- If $b_u = 1$ holds, then d_u refers to the initial degree of u in the input graph.

When receiving data item $\langle u, d_u, 0, v \rangle$ the degree of u in the input graph may be larger than the current degree d_u ; when receiving data item $\langle u, d_u, 1, v \rangle$ the current degree of u might be smaller than the original degree d_u (but the current degree is at least one, since u is incident with v).²⁶

Discussion. Compared with \mathcal{DS} , class \mathcal{DS}_{01} also contains the STATIC-MINGREEDY algorithm, see Lemma 56 below. More generally, using the original degree, an algorithm can e.g. prefer a node whose degree is small not because it dropped during the course of the algorithm but because it was small from begin on. This is impossible for \mathcal{DS} -algorithms.

We show an inapproximability bound for \mathcal{DS}_{01} which matches the bound for the more restricted class \mathcal{DS} presented in [BW15].

²⁶During an execution of an algorithm in \mathcal{DS}_{01} , unseen data items with bit $b_u = 1$ remain unchanged until being removed from set U (since the original degree of a node remains the same), whereas unseen data items with $b_u = 0$ are updated as discussed after Definition 52.

- ii. For the class \mathcal{VA} (cf. the *vertex adjacency formulation* in [BBLM10] or the *node model* in [DI09]) a data item

$$\langle u, v_1, \dots, v_{D_u}, v \rangle$$

states a node u along with all neighbors v_1, \dots, v_{D_u} of u in the input graph, i.e. the degree of u in the input graph is D_u . Furthermore, the data item contains a distinguished neighbor $v \in \{v_1, \dots, v_{D_u}\}$ to be matched with u .²⁷

Discussion. Using this type of data items, an algorithm A is able to detect neighbor relationships between nodes: in any round algorithm A may remember all node names processed so far, hence A can give high priority to nodes adjacent with nodes processed in previous rounds. Selecting neighbors of previously matched nodes allows A to “traverse” the graph (e.g. to construct alternating paths), which is not possible for algorithms in \mathcal{DS}_{01} . Moreover, these data items allow to infer distance information between nodes. E.g. given a data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ an algorithm can conclude that nodes v_1 and v_2 are at distance at most two to each other. Such distance information is not exposed for algorithms in \mathcal{DS}_{01} .

Furthermore, recall that an algorithm in \mathcal{DS}_{01} may specify the current degree of node u or the degree of u in the input graph, but not both. As we show next, a \mathcal{VA} -algorithm is able to specify both degrees, since it can infer the current degree of a node from its data item. (In particular, data items are not updated as e.g. in class \mathcal{DS}_{01} in order to reflect current degrees.)

Lemma 54. *Assume that a \mathcal{VA} -algorithm receives data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$. Node u has current degree $D_u - m_u$, where m_u is the number of those nodes of v_1, \dots, v_{D_u} which have already been matched in previous rounds.*

Proof. We have to show that in the reduced graph each neighbor v_i of u in the input graph is either already matched or still adjacent with u . This is a consequence of the following two facts. First, node v_i cannot already be matched and still be adjacent with u at the same time. Secondly, if v_i is not yet matched and not adjacent with u anymore, then we obtain a contradiction, since also u is not yet matched and edge $\{u, v\}$ is still contained in the graph. \square

²⁷Note that in [BBLM10] the vertex adjacency formulation is defined slightly different. There, a data item $\langle u, v_1, \dots, v_{D_u} \rangle$ does not contain a distinguished neighbor v but only the list of neighbors of u . Therefore, when receiving a data item, a matching algorithm has to select one of the neighbors of u first and thereafter has to pick the edge connecting both nodes.

Our formulation is equivalent, since the choice of the neighbor $v \in \{v_1, \dots, v_{D_u}\}$ of u for data item $\langle u, v_1, \dots, v_{D_u} \rangle$ is deterministic: the data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ can be “precomputed”.

To verify the inclusion relations in Lemma 56 below (and the remaining inclusion relations claimed in Figure 29) we use standard simulation arguments. Specifically, for each pair of classes their definitions are the same except for the data items used. Therefore it suffices to provide a suitable transformation of a priority order.

Lemma 55. *Let \mathcal{A} and \mathcal{B} be classes of adaptive priority algorithms for the matching problem which differ only in the type of data items used.*

So show that $\mathcal{A} \subseteq \mathcal{B}$ holds it suffices to prove that for every algorithm $A \in \mathcal{A}$ there is an algorithm $B \in \mathcal{B}$ such that the following holds for each graph and every round:

Algorithm A submits a priority order π_A and receives a data item $\langle u, \dots, v \rangle$ if and only if B computes a transformation π_B of π_A such that B receives a data item $\langle u, \dots, v \rangle$, where dots indicate additional information on nodes u and v specific to class \mathcal{A} resp. \mathcal{B} .

Proof. In each round algorithm B picks the same edge as A . □

Lemma 56. *We have*

$$\begin{aligned} \text{STATIC-MINGREEDY} \in \mathcal{DS}_{01} & \qquad \qquad \qquad \text{as well as} \\ \mathcal{DS} \subseteq \mathcal{DS}_{01} \subseteq \mathcal{WA}. \end{aligned}$$

Proof. We show that **STATIC-MINGREEDY** $\in \mathcal{DS}_{01}$ holds. Recall that the algorithm begins with computing an arbitrary order on nodes which is sorted ascending by degree (in the input graph) and then processes nodes in this order. For each non-isolated node **STATIC-MINGREEDY** picks an arbitrary incident edge (and then removes all edges incident with both matched nodes).

To implement **STATIC-MINGREEDY** as a \mathcal{DS}_{01} -algorithm, we compute the priority order $\pi = \langle *, 1, 1, * \rangle, \langle *, 2, 1, * \rangle, \langle *, 3, 1, * \rangle, \dots, \langle *, |V| - 1, 1, * \rangle$ before the first round: we denote by $\langle *, d, 1, * \rangle$ a (total) order on “edges” with a node of degree d in the input graph (the bit is set to 1), where we assume that $\langle *, d, 1, * \rangle$ is ordered lexicographically by node names. Order π is submitted in each round, i.e. we receive a data item $\langle u, d, 1, v \rangle$ of a non-isolated node u with smallest possible degree d in the input graph, i.e. we pick an arbitrary edge incident with u .

We show that $\mathcal{DS} \subseteq \mathcal{DS}_{01}$ holds. Let $A \in \mathcal{DS}$. We have to show that there is an algorithm $A' \in \mathcal{DS}_{01}$ which simulates A . By Lemma 55 it suffices to give a transformation of a priority order π of A into a priority order π' of A' such that both algorithms receive data items $\langle u, \dots, v \rangle$ for the same edge $\{u, v\}$.

We replace each data item $\langle u, d_u, v \rangle$ in π with $\langle u, d_u, 0, v \rangle$ in π' . All data items in π as well as in π' refer to current degrees in the reduced graph. Thus both algorithms receive a data item for the same edge.

To show that $\mathcal{DS}_{01} \subseteq \mathcal{VA}$ holds, let $A \in \mathcal{DS}_{01}$. We have to show that there is an algorithm $A' \in \mathcal{VA}$ which simulates A . In each round, algorithm A' transforms the priority order π of A by replacing each data item $\langle u, d_u, b_u, v \rangle$ with several consecutive data items, depending on bit b_u .

- First, assume that $b_u = 1$ holds, i.e. that the degree of u in the input graph is $D_u = d_u$. Recall that a \mathcal{VA} -data item for node u contains all D_u neighbors in the input graph. Algorithm A' replaces $\langle u, d_u, b_u, v \rangle$ with several consecutive data items, namely one data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ for all possible sets $\{v_1, \dots, v_{D_u}\}$ of size D_u which contain v .

Observe that a data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ is contained in the graph if and only if there is a node u of degree $d_u = D_u$ in the input graph which is adjacent with v .

- Now assume that we have $b_u = 0$, i.e. that d_u is the current degree of u in the reduced graph. As a consequence of Lemma 54, when inserting a data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ into a priority order, algorithm A' can request current degree d_u of node u by choosing $D_u - d_u$ neighbors of u as already matched nodes (the set of already matched nodes is tracked throughout the course of the algorithm).

How does A' transform a data item $\langle u, d_u, b_u, v \rangle$ with $b_u = 0$? The degree D_u of u in the input graph is $d_u \leq D_u \leq |V| - 1$, where V denotes the set of all nodes. Algorithm A' replaces $\langle u, d_u, b_u, v \rangle$ with several consecutive data items, namely one data item $\langle u, v_1, \dots, v_{d_u}, v_{d_u+1}, \dots, v_{D_u}, v \rangle$ for each possible choice

- of the initial degree D_u of u in the input graph,
- of d_u many nodes v_1, \dots, v_{d_u} adjacent to u (where $v = v_i$ for some i), and
- of $D_u - d_u$ many nodes $v_{d_u+1}, \dots, v_{D_u}$ which are already matched.

Observe that a data item $\langle u, v_1, \dots, v_{d_u}, v_{d_u+1}, \dots, v_{D_u}, v \rangle$ is contained in the input if and only if there is a node u of current degree d_u which is adjacent with v .

Combining both cases, algorithm A' receives a data item from the replacements for data item $\langle u, d_u, b_u, v \rangle$ if and only if A receives that data item. \square

5.2.2 Degree Sensitivity for Multiple Nodes

Again, we only change the definition of data items and leave the rest of the Definition 52 unchanged.

- iii. For the class \mathcal{DSE}_{01} a data item exposes degree information on both nodes of an edge. In particular, a data item

$$\langle (u, d_u, b_u), (v, d_v, b_v) \rangle$$

contains the same information as for \mathcal{DS}_{01} and additionally states the degree d_v of node v and a bit b_v which decides whether d_v is the degree of v in the reduced graph ($b_v = 0$) or in the input graph ($b_v = 1$).

Discussion. As we point out in Lemma 57, algorithms DOUBLE-MINGREEDY and MDS belong to \mathcal{DSE}_{01} . However, both algorithms are not contained in \mathcal{DS} , since to select the next edge DOUBLE-MINGREEDY and MDS incorporate degree information of both nodes.

Lemma 57. *We have*

$$\begin{aligned} \text{MDS} &\in \mathcal{DSE}_{01}, \\ \text{DOUBLE-MINGREEDY} &\in \mathcal{DSE}_{01}, && \text{as well as} \\ \mathcal{DS}_{01} &\subseteq \mathcal{DSE}_{01}. \end{aligned}$$

Proof. To implement **MDS** as a \mathcal{DSE}_{01} -algorithm, in each round the priority order contains all possible data items $\langle (u, d_u, 0), (v, d_v, 0) \rangle$ sorted ascending by current degree sum $d_u + d_v$.

To implement **DOUBLE-MINGREEDY** as a \mathcal{DSE}_{01} -algorithm, in each round the priority order contains all possible data items $\langle (u, d_u, 0), (v, d_v, 0) \rangle$ sorted lexicographically by (d_u, d_v) , i.e. such that for any given d_u all data items of current degree d_u are next to each other within the same “block”, blocks are sorted ascending by d_u , and within each block data items are sorted ascending by the current degree d_v of v .

To prove that $\mathcal{DS}_{01} \subseteq \mathcal{DSE}_{01}$ holds, we show that an algorithm $A \in \mathcal{DS}_{01}$ can be simulated by an algorithm $A' \in \mathcal{DSE}_2$. Since \mathcal{DS}_{01} and \mathcal{DSE}_{01} differ only the definition of data items, we only have to show how A' transforms a priority order computed by A into a priority order of “double degree sensitive” data items, cf. Lemma 55. Each data item $\langle u, d_u, b_u, v \rangle$ is replaced with several consecutive data items, namely

one data item $\langle (u, d_u, b_u), (v, d_v, b_v) \rangle$ for all possible choices of d_v and b_v . A data item $\langle (u, d_u, b_u), (v, d_v, b_v) \rangle$ exists in the input if and only if the graph contains a node u of degree d_u according to b_u with neighbor v . Consequently, algorithm A' picks the same edge as A . \square

Picking Paths. All algorithms defined so far may pick only a single edge at a time. As we will see, there are instances for which they perform poorly. This gives rise to the question whether better approximation performance can be achieved by picking several edges at a time. As before, we build upon Definition 52 by modifying the type of data items (only).

iv. Let $k \geq 1$ be a constant integer. For the class \mathcal{DSP}_{01} a data item

$$\langle P_1, \dots, P_k \rangle$$

contains k paths, where a path

$$P_i = ((v_j^i, d_j^i, b_j^i))_{j=1}^{n_i}$$

is a sequence of n_i nodes v_j^i together with their degree d_j^i and a bit b_j^i deciding whether d_j^i is the degree of v_j^i in the reduced graph ($b_j^i = 0$) or in the input graph ($b_j^i = 1$).

In a data item $\langle P_1, \dots, P_k \rangle$ received by an algorithm node v_j^i has degree d_j^i in the reduced graph or in the input graph, depending on b_j^i . Moreover, for path P_i the edges

$$\{v_1^i, v_2^i\}, \{v_2^i, v_3^i\}, \dots, \{v_{n_i-1}^i, v_{n_i}^i\}$$

are contained in the reduced graph and the algorithm picks edges

$$\{v_1^i, v_2^i\}, \{v_3^i, v_4^i\}, \dots, \{v_{n_i-1}^i, v_{n_i}^i\},$$

where we assume that n_i is an even non-negative integer.

We demand that the total number of edges being picked (in all paths) is at most $\frac{1}{2} \sum_{r=1}^k n_r \leq k$. Hence, if a path P_i contains more than one edge, then another path P_j must be empty.

Discussion. Why is the number of edges per round bounded by k ? Allowing to pick an arbitrary number of edges enables an algorithm to prefer sets of $\lfloor \frac{|V|}{2} \rfloor$ edges—which constitute perfect matchings—over sets of $\lfloor \frac{|V|}{2} \rfloor - 1$ edges, over sets of $\lfloor \frac{|V|}{2} \rfloor - 2$ edges, etc. Therefore an algorithm can compute a maximum matching in only one round. This definition would be too liberal to model the greedy construction of a matching.

Lemma 58. *We have*

$$\mathcal{DSE}_{01} \subseteq \mathcal{DSP}_{01}.$$

Proof. Let $A \in \mathcal{DSE}_{01}$. We have to show that there is an algorithm $A' \in \mathcal{DSP}_{01}$ which simulates A . By Lemma 55 we only have to provide a transformation of a priority order π of A . In each round, algorithm A' transforms π by replacing each data item $\langle (u, d_u, b_u), (v, d_v, b_v) \rangle$ with a data item $\langle P_1, \dots, P_k \rangle$ in which paths P_2, \dots, P_k contain $n_2 = \dots = n_k = 0$ edges and we have $P_1 = (u, d_u, b_u), (v, d_v, b_v)$. \square

5.2.3 Degree Ignorance

We introduce an additional modification of Definition 52. Using the new class we prove in Section 5.3 that a greedy matching algorithm must select nodes of small degree in order to achieve a non-trivial approximation guarantee.

- v. For the class \mathcal{DI} of *degree ignorant* algorithms a data item

$$\langle u, N_u, v, N_v \rangle,$$

specifies two nodes u and v and sets of nodes N_u and N_v . If the algorithm receives data item $\langle u, N_u, v, N_v \rangle$, then for $w \in \{u, v\}$ each node in N_w is a neighbor of w in the reduced graph. However, not every neighbor of w has to be contained in N_w : if $N(w)$ is the complete set of neighbors of w in the input graph, then we only demand that $N_w \subseteq N(w)$ holds. For convenience, we assume $u \in N_v$ and $v \in N_u$.

Discussion. In each data item, an algorithm in \mathcal{DI} may specify arbitrarily detailed requirements about the sets of neighbors of *both* matched nodes. This sets \mathcal{DI} apart from \mathcal{VA} , for which requirements for only one node can be formulated, or from \mathcal{DSE}_{01} , for which only degrees but no neighbor names can be specified. (In particular, note that assuming $N_w = N(w)$ for $w \in \{u, v\}$ would imply $\mathcal{VA} \subseteq \mathcal{DI}$ as well as $\mathcal{DSE}_{01} \subseteq \mathcal{DI}$.)

As we show as part of the next result, common algorithms which request small degree nodes are not members of \mathcal{DI} .

Lemma 59. *We have*

$$\mathcal{QC} \subseteq \mathcal{DI}.$$

Algorithms MINGREEDY, 1-2-GREEDY, KARPSIPSER, DOUBLE-MINGREEDY, *and* MDS *are not contained in* \mathcal{DI} .

Proof. The proof that $\mathcal{QC} \subseteq \mathcal{DI}$ holds proceeds analogously to the proof of $\mathcal{QC} \subseteq \mathcal{DS}$ in Lemma 53. In particular, the simulation of an algorithm $A \in \mathcal{QC}$ by an algorithm $A' \in \mathcal{DI}$ proceeds like follows. In each round, algorithm A' simulates a sequence of probes for edges which ends with a successful probe. After A' has picked the same edge as A , the simulation is continued in the next round of A' . To obtain the probe order for one round, algorithm A' simulates A under the assumption that no probe is successful: this allows to correctly simulate the behavior of A on the reduced graph. The obtained probe order $\{u, v\}, \{w, x\}, \dots$ is transformed into the priority order $\langle u, *, v, * \rangle, \langle w, *, x, * \rangle, \dots$ where for probe $\{y, z\}$ the sub-order $\langle y, *, z, * \rangle$ contains data items for all possible choices of sets of neighbors in the $*$ -ed positions. If a probe for edge $\{y, z\}$ is not successful, then no data item in $\langle y, *, z, * \rangle$ is contained in the input. If A successfully probes for edge $\{y, z\}$, then one of the data items in $\langle y, *, z, * \rangle$ is contained in the input and received by A' . Thus A' picks the same edge as A .

To prove the second statement, we argue that an algorithm $A \in \mathcal{DI}$ is unable to request nodes of small degree. In particular, we show that in the first round algorithm A cannot request a node u of small degree d in the input graph. Why? Since therefore in a data item $\langle u, N_u, v, N_v \rangle$ the set N_u of neighbors of u must be of size at most d . But the definition of \mathcal{DI} only requires $N_u \subseteq N(u)$ for the set $N(u)$ of neighbors of u in the input graph: thus we might have $|N(u)| > d$ for the received data item, i.e. node u has larger degree in the input graph than d . \square

5.2.4 Hypergraph Matching

To study limits of greedy algorithms for the k -Hypergraph Matching Problem (cf. page 30), we generalize \mathcal{WA} to the class \mathcal{WA}^k , for which we use data items of the form

$$\langle u, W_1, \dots, W_{D_u}, W_i \rangle.$$

Each W_j is a set of $k - 1$ nodes and $W_i \in \{W_1, \dots, W_{D_u}\}$ holds. If an algorithm receives data item $\langle u, W_1, \dots, W_{D_u}, W_i \rangle$, then u has D_u incident edges in the input graph, namely $\{u\} \cup W_1, \dots, \{u\} \cup W_{D_u}$, and edge $\{u\} \cup W_i$ must be picked.

Lemma 60. *We have*

$$\mathcal{WA} \subseteq \mathcal{WA}^k.$$

Proof. Observe that k -dimensional hypergraph matching becomes the standard matching problem for $k = 2$. In particular, consider data item $\langle u, W_1, \dots, W_{D_u}, W_i \rangle$. For $k = 2$ an “edge” W_i is a set of size $|W_i| = 1$. Therefore $\langle u, W_1, \dots, W_{D_u}, W_i \rangle$ is merely another notation of data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ for class \mathcal{WA} . The simulation of a \mathcal{WA} -algorithm by a \mathcal{WA}^k -algorithm is now straight-forward. \square

5.3 Inapproximability Results

For the classes of adaptive priority algorithms defined in Section 5.2 we show the inapproximability bounds given in Table 3. We consider bipartite and non-bipartite graphs as well as k -uniform hypergraphs (in which every edge contains k nodes). We frequently refer to the largest degree of a node in the graph, which is either a constant (top rows), given as a variable Δ (middle rows), or we assume arbitrarily large degrees (bottom rows). (Note that bounds for empty cells are implied from bounds in cells above and to the left.) First, we give an overview over the results and discuss implications.

Table 3: Inapproximability bounds by graph type (columns), maximum degree (rows), and type of algorithm (classes are shaded like in Figure 29)

		bipartite graphs	general graphs	k -uniform hypergraphs
constant	2	$\mathcal{DI} : \frac{1}{2} \quad (15)$		
	3	$\mathcal{DSE}_{01} : \frac{3}{4} \quad (16)$		
	4			$\mathcal{WA}^k : \frac{1}{k} \quad (17)$
bounded	Δ	$\mathcal{DS}_{01} : \frac{\Delta}{2\Delta-2} \quad (18)$	$\mathcal{WA} : \frac{\Delta-1}{2\Delta-3} \quad (19)$	
		$\mathcal{MDS} : \frac{\Delta}{2\Delta-2} \quad (20)$		
		$\mathcal{DSE}_{01} : \frac{\Delta+1}{2\Delta-2} \quad (21)$		
large	$\Delta \rightarrow \infty$	$\mathcal{DSR}_{01} : \frac{1}{2} \quad (22)$		
		$\mathcal{WA} : \frac{3}{4} \quad (23)$		

Maximum Cardinality Matching. We study \mathcal{WA} -algorithms for general and for bipartite graphs. By (19), on general degree bounded graphs no \mathcal{WA} -algorithm achieves better approximation ratio than 1-2-GREEDY, cf. our performance guarantees in Theorems 15 and 21. Consequently, the ability to traverse the graph gives no advantage over the ability to exploit degree information. Our construction crucially relies on the existence of triangles in the graph.

Since triangles do not exist in bipartite graphs, tailoring our construction to the bipartite scenario yields only weaker bounds. In particular, if degrees are unbounded then by (23) no \mathcal{WA} -algorithm performs better than factor $\frac{3}{4}$. Whether there exists a \mathcal{WA} -algorithm achieving approximation factor larger than $\frac{1}{2}$ remains an open question.

Bipartite Graphs. We analyze the worst case performance of the remaining classes on bipartite graphs. By (15), algorithms in \mathcal{DI} , which are able to exploit neighbor relationships but are unable to request nodes of small degree, fail to achieve a non-trivial approximation guarantee, even on paths. Consequently, to obtain a non-trivial approximation guarantee, a greedy matching algorithm must incorporate nodes of small degree into the choice of the next edge.

Thus we compare with the approximation performance of the KARPSIPSER algorithm. If degrees are bounded by Δ , then (18) implies that \mathcal{DS}_{01} -algorithms perform no better than the KARPSIPSER algorithm, cf. our performance guarantee in Theorem 35, even though they are equipped with the additional capability to access node degrees in the input graph.

Moreover, the MDS algorithm does not perform better than KARPSIPSER, even though it uses degree information on both nodes of the next edge, as (20) shows. By (16), if degrees are bounded by at most $\Delta = 3$, then no \mathcal{DSE}_{01} -algorithm performs better than KARPSIPSER, since $\frac{\Delta}{2\Delta-2} = \frac{3}{4}$ holds. The class \mathcal{DSE}_{01} also contains the DOUBLE-MINGREEDY algorithm.

We conjecture that no \mathcal{DSE}_{01} -algorithm achieves better performance than KARPSIPSER, namely factor $\frac{\Delta}{\Delta-2}$ for arbitrary Δ . If the conjecture holds, then utilizing degree information for two nodes instead of only one does not yield a benefit for bipartite graphs. To support our conjecture we show in (21) that \mathcal{DSE}_{01} -algorithms perform only scarcely better than KARPSIPSER, namely with factor at most $\frac{\Delta+1}{2\Delta-2}$.

Algorithms in \mathcal{DSP}_{01} may specify the same information on each node as \mathcal{DSE}_{01} -algorithms, but instead of being restricted to picking one edge in each step they may pick multiple edges (arranged in alternating paths) at the same time. However, even

this power does not allow for a fundamental change in approximation quality: for large graphs the trivial guarantee $\frac{1}{2}$ cannot be beaten, as (22) shows.

Altogether, our results show that among greedy matching algorithms achieving non-trivial approximation guarantees 1-2-GREEDY and the KARPSSIPER algorithm stand out as an exceptional combination of conceptual simplicity and performance.

k -Hypergraph Matching. For a complete picture we also study the greedy approach when applied to the k -Hypergraph Matching Problem. As it turns out, factor $\frac{1}{k}$ cannot be surpassed by \mathcal{WA}^k -algorithm, as (17) shows. This performance is worst possible, since any maximal matching in a k -uniform hypergraph has size at least $\frac{1}{k}$ times optimal (each picked edge touches at most k optimal edges). Thus the success of greedy algorithms for matching does not carry over to hypergraph matching.

5.3.1 Proof Techniques

To verify the inapproximability bounds claimed in Table 3, we analyze an algorithm A in the *adaptive priority game* against an adversary B . Adversary B has to play a *consistent* game, i.e. given the final construction as input to A (without an adversary being involved) must result in the same sequence of submitted priority orders, received data items, and decisions and hence produce the same solution.

The fundamental idea is to force algorithm A to perform $\frac{1}{2}$ -rounds: in a $\frac{1}{2}$ -round two edges of a maximum matching are removed from the graph. Since algorithm A picks only one edge, a large number of $\frac{1}{2}$ -rounds implies that the approximation ratio tends to $\frac{1}{2}$. Our constructions vary in *how* a $\frac{1}{2}$ -round is enforced. Before we present details, we discuss adversarial strategies common to algorithms of various classes and how these strategies relate to one another. The following overview is arranged (roughly) from simple to complex strategies.

- For algorithms in classes \mathcal{DI} and \mathcal{WA}^k consistency of B is easiest to achieve.

Fixed Construction. For either class, the *structure* of the graph does not depend on the actions of algorithm A , i.e. constructions for different algorithms yield isomorphic graphs. Node labels are chosen by B depending on A . In particular, from the first priority order submitted by A adversary B presents the highest priority data item i with a degree present in the graph and establishes consistency by labeling nodes according to the names given in i .

A Game of one Round. The construction asserts that the edge picked by A is a poor choice. In particular, the first round turns out to be a $\frac{1}{2}$ -round (for \mathcal{DI}) resp. a $\frac{1}{k}$ -round (for \mathcal{VA}^k). Adversary B ends the game after the first round, hence consistency does not have to be maintained over subsequent rounds.

- A bound for \mathcal{DS} is implied from our construction for \mathcal{DS}_{01} , which is closely related with the construction for \mathcal{DSE}_{01} . Both are more complex than the constructions for \mathcal{DI} and \mathcal{VA}^k in the following two ways. First, consistency of B must be maintained over more than one round. Secondly, constructions for different algorithms are not isomorphic graphs, but the set of edges depends on the actions taken by algorithm A . However, all constructions share an isomorphic subgraph (the “skeleton” or “backbone”) to which edges are added depending on A .

Allowed Degrees. Adversary B controls the game by limiting the possible moves of algorithm A . Therefore B makes use of a set D of *allowed* degrees: in the input graph G_1 the degree of each node w satisfies $d_1(w) \in D$. In particular, we have $1 \notin D$ to prevent A from performing optimally by picking an edge incident with a node of degree 1.

A $\frac{1}{2}$ -round t occurs when also the reduced graph G_t contains only allowed degrees, i.e. each node w has a degree $d_t(w) \in D$. (Observe that this is the case in the first round.)

How does the construction of B proceed in a $\frac{1}{2}$ -round t ? Among data items with degrees in D in the order submitted in round t , it is consistent to present the one with highest priority (and relabel nodes accordingly). Assume that data item i is presented and that in i node w has degree d . The construction asserts that node w has degree $d = d_1(w) = d_t(w)$ in both the input graph as well as in the reduced graph. This is consistent with any value of the degree bits given in i .

Fragments of the Reduced Graph. If in round t the reduced graph G_t contains a node w of degree $d_t(w) \notin D$, then no $\frac{1}{2}$ -round can be enforced. Has B lost control over the game? No, since the construction of B asserts that node w belongs to a connected component C of G_t with the following crucial property:

algorithm A may pick an *optimal* number of edges in C *without* surpassing the desired inapproximability bound.

Edges in C are picked independently of the rest of the game: the restricted type of data items asserts that, first, during previous rounds algorithm A could not gather

knowledge about neighbors of already matched nodes, in particular about such neighbors in C , and, secondly, algorithm A is unable to do so in future rounds. Hence we may w.l.o.g. assume that A picks edges in C instantly, and we proceed with the adversarial construction after all edges in C are removed. At this time the construction of B asserts that, again, all current degrees belong to set D , i.e. the next $\frac{1}{2}$ -round occurs.

- For \mathcal{VA} -algorithms, the adversary has to take into account that an algorithm is able to detect neighbor relationships. In order to maintain consistency, adversary B has to proceed more carefully and therefore introduces the following modifications to the “game plan”.

The game begins with a large number of rounds in which algorithm A does not solve any fragments and each round is a $\frac{1}{2}$ -round (again, a $\frac{1}{2}$ -round t occurs when the reduced graph G_t contains only the same allowed degrees as the input graph G_1). Adversary B keeps up the following crucial invariant during each of these rounds.

All nodes which occurred in a data item presented in an earlier round *cannot* be matched.

(Observe that the invariant holds in the first round.) Thus it is consistent of B to present the data item of a node which was not processed before—in particular, the highest priority such data item with an allowed degree in the submitted order. Since A could not gather knowledge about the received node, adversary B has no commitments to satisfy and thus has sufficient freedom to enforce the next $\frac{1}{2}$ -round while keeping up the invariant.

As soon as the adversary cannot keep up the invariant and cannot enforce further $\frac{1}{2}$ -rounds, the algorithm is free to solve the remaining fragment of the graph optimally.

5.3.2 Proofs

An overview of the proof order is given in Table 4.

Theorem 61. *Let $A \in \mathcal{DI}$ be a degree ignorant algorithm. There is an input graph for which A computes a matching of size exactly $\frac{1}{2}$ times optimal. In particular, this input graph is a path on four nodes.*

Proof. We describe the adaptive priority game of A against an adversary B . Since the structure of the hard instance is fixed, only node labels are chosen depending on the

Table 4: Proof order of inapproximability results (related groups of related proofs are separated by double horizontal lines)

Result (cf. Table 3)	Statement	Page
(15) \mathcal{DI}	Theorem 61	129
(18) \mathcal{DS}_{01}	Theorem 62	130
(19) \mathcal{VA} (general graphs)	Lemma 63 Theorem 64	135 139
(23) \mathcal{VA} (bipartite graphs)	Theorem 65	142
(20) MDS	Theorem 66	145
(16) \mathcal{DSE}_{01} ($\Delta = 3$)	Theorem 67	145
(21) \mathcal{DSE}_{01} ($\Delta \geq 4$)	Theorem 68	147
(22) \mathcal{DSR}_{01}	Theorem 69	149
(17) \mathcal{VA}^k (k -uniform hypergraphs)	Theorem 70	151

actions taken by A . From the first priority order submitted by A , adversary B presents the highest priority data item $\langle u, N_u, v, N_v \rangle$ with $1 \leq |N_u| \leq 2$ and $1 \leq |N_v| \leq 2$. (Recall that we have $u \in N_v$ as well as $v \in N_u$.) Then B relabels nodes in the path such that $\{u, v\}$ is the middle edge. This is consistent, since the middle nodes have degrees at least one and at most two. In particular, since A picks the middle edge, algorithm A cannot pick the edges incident with the degree-1 nodes of the path, which constitute a maximum matching of size two. \square

Note. The same construction allows to show the same bound if an algorithm may additionally choose the partition of each node.

Degree Sensitive Algorithms. First, we discuss “one-sided” degree sensitive algorithms, like e.g. KARPSSIPSE or MINGREEDY.

Theorem 62. *Let $A \in \mathcal{DS}_{01}$. For any $\Delta \geq 3$ and $\varepsilon > 0$, there is a bipartite input graph of degree at most Δ (and with a perfect matching) for which A computes a matching of size at most $\frac{\Delta}{\Delta-2} + \varepsilon$ times optimal.*

Proof. First, we prove the result for graphs of degree at most $\Delta \geq 4$ and without a perfect matching. Thereafter, we modify the construction such that the statement also holds for $\Delta = 3$ and a perfect matching.

We describe the adaptive priority game between A and an adversary B , who processes priority orders submitted by A in order to construct a hard input instance. Adversary B constructs a graph which contains k traps T_1, T_2, \dots, T_k , cf. Figure 30. Trap T_i contains

a *left cycle* on nodes $c_1^i, c_2^i, c_3^i, c_4^i$ which is connected via an edge $\{c_1^i, p_1^i\}$ to a *left path* on nodes p_1^i, p_2^i . Trap T_i also contains a *right cycle* on nodes $d_1^i, d_2^i, d_3^i, d_4^i$ connected via $\{d_1^i, q_1^i\}$ to a *right path* on nodes q_1^i, q_2^i . The left path is connected to the right cycle via edges $\{p_2^i, d_1^i\}, \{p_2^i, d_3^i\}$, and analogously the right path of T_i is connected to the left cycle of the next trap T_{i+1} via edges $\{q_2^i, c_1^{i+1}\}, \{q_2^i, c_3^{i+1}\}$; the right path of the last trap T_k is connected to an extra cycle on nodes e_1, e_2, e_3, e_4 via edges $\{q_2^k, e_1\}, \{q_2^k, e_3\}$; an extra node e_0 connects to the left cycle nodes c_1^1, c_3^1 of the first trap. The left and right cycles in T_i are connected by $\Lambda = \Delta - 4$ many length-three paths on nodes $w_j^i, x_j^i, y_j^i, z_j^i$ via edges $\{c_1^i, w_j^i\}, \{c_3^i, w_j^i\}$ and $\{z_j^i, d_1^i\}, \{z_j^i, d_3^i\}$ for $1 \leq j \leq \Lambda$.

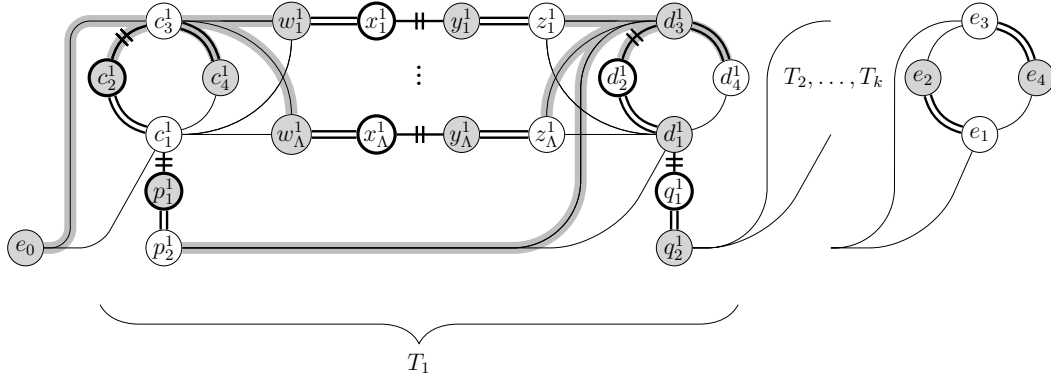


Figure 30: The construction of adversary B : algorithm A receives data items for bold nodes (partitions are marked with white and gray nodes, gray edges form fragments of the graph in rounds $\Lambda+2$ resp. $\Lambda+4$ of the adaptive priority game)

This constitutes the “backbone” of the construction, to which B adds further edges during the game depending on the actions taken by A . The game proceeds as follows. For each trap T_i algorithm A picks Δ edges to include in the solution (crossed edges in Figure 30), whereas T_i contains $2\Delta - 2$ edges of a maximum matching (double edges). Furthermore, only a constant number of edges incident with nodes e_0, \dots, e_4 can be scored by A . Hence for large k algorithm A achieves approximation ratio at most $\frac{\Delta}{2\Delta - 2} + \varepsilon$.

To start the game, adversary B announces the number $k \cdot (12 + 4\Lambda) + 5$ of nodes. The construction of B proceeds such that after the first Δ rounds node e_0 and all nodes in T_1 but q_2^1 are isolated. The graph to be constructed thereafter is one trap “shorter” with q_2^1 instead of e_0 connected to the leftmost trap. Adversary B repeats its strategy for T_2, T_3, \dots, T_k . After all edges in T_k are removed, algorithm A scores at most two edges in the “fragment” consisting of nodes $q_2^k, e_1, e_2, e_3, e_4$ and edges connecting these nodes, cf. Section 5.3.1.

We discuss the first $\Lambda = \Delta - 3$ rounds. Observe that in the first round the minimum degree is two. Adversary B chooses the set

$$D = \{2, \dots, \Delta\}$$

of allowed degrees, cf. Section 5.3.1. In each of rounds $1 \leq j \leq \Lambda$, adversary B presents the highest priority data item $\langle u, d_u, b_u, v \rangle$ with $d_u \in D$ in the respective priority order submitted by A , ignoring the value of bit b_u . Adversary B then relabels nodes in the graph such that $u = x_j^1$ and $v = y_j^1$ holds, i.e. algorithm A picks the crossed edges in the length-three paths.

In each round B has committed to node $u = x_j^1$ having degree d_u , either in the reduced graph or in the input graph depending on bit b_u . To act consistently, adversary B asserts that both degrees are equal to d_u . How? Degree d_u might be larger than two, therefore B inserts additional edges incident with x_j^1 into the graph in Figure 30. The $d_u - 2$ additional edges connect x_j^1 with arbitrary nodes in the set

$$\{w_1^1, \dots, w_\Lambda^1, c_2^1, c_4^1, q_2^1\} \setminus \{w_j^1\}.$$

This set has cardinality $\Lambda + 3 - 1 = \Delta - 4 + 2 = \Delta - 2 \geq d_u - 2$ and only contains nodes outside the partition of x_j^1 .

The additional edges are consistent: in previous rounds A could not gather knowledge about the neighborhood of $u = x_j^1$, or any other still unmatched node, therefore the additional edges do not have effect on previous actions taken by A .

Edges incident with u and v —including additional edges—are removed from the graph, hence in round $j + 1$ the minimum degree is two, again.

The degrees of nodes receiving additional edges are increased to at most

$$\Delta - 1 \tag{24}$$

during rounds $1 \leq j \leq \Lambda$: the w_j^1 have degree at most $3 + \Lambda - 1 = \Delta - 2$, both c_2^1 and c_4^1 have degree at most $2 + \Lambda = \Delta - 2$, and q_2^1 has degree at most $3 + \Lambda = \Delta - 1$.

After rounds $1, \dots, \Lambda$, recall that the minimum degree is two. In round $\Lambda + 1$ adversary B again presents the highest priority item $\langle u, d_u, b_u, v \rangle$ with $2 \leq d_u \leq \Delta$ in the submitted order. This time B relabels nodes such that $u = p_1^1$ and $v = c_1^1$ hold (hence A

picks the crossed edge connecting the left cycle and path), and inserts $d_u - 2 \leq \Delta - 2$ additional edges connecting u with arbitrary nodes in the set

$$\{z_1^1, \dots, z_\Lambda^1, d_2^1, d_4^1\}.$$

The degrees of nodes receiving an additional edge are increased by only one, i.e. they do not exceed $4 \leq \Delta$.

In round $\Lambda+2$ a star centered at node c_3^1 forms a fragment of the graph. Since A computes a maximal matching, these star nodes become isolated when A matches c_3^1 . W.l.o.g. we assume that A isolates these nodes in round $\Lambda+2$. Observe here, that for the selected node the current and original degree might in fact be different, however, algorithm A cannot take any advantage since at most one edge can be picked in the star centered at c_3^1 .

Similarly, the game proceeds for the right cycle and path. In round $\Lambda+3$, algorithm A matches nodes $u=q_1^1$ and $v=d_1^1$, where additional $d_u - 2$ edges connect q_1^1 with arbitrary nodes in the set

$$\{w_1^2, \dots, w_\Lambda^2, c_2^2, c_4^2\} \tag{25}$$

of left path and cycle nodes in trap T_2 . In round $\Lambda+4=\Delta$ a star centered at d_3^1 forms a fragment of the graph. W.l.o.g. again, algorithm A scores one edge in this round, when isolating all nodes in this fragment.

Adversary B repeats its strategy for trap T_2 . We have to pay attention to the following subtlety. Adversary B might previously have constructed edges connecting some of nodes $w_1^2, \dots, w_\Lambda^2, c_2^2, c_4^2$ with q_1^1 , cf. (25). Therefore, additional edges in T_2 might increase the degrees of these nodes up to Δ , instead of up to $\Delta - 1$, cf. (24). This applies analogously to T_3, T_4, \dots, T_k .

$\Delta = 3$. We modify the above construction as illustrated in Figure 31. Paths on nodes $w_j^i, x_j^i, y_j^i, z_j^i$ and edges incident with these nodes do not exist. Left and right paths now have four nodes p_1^i, \dots, p_4^i resp. q_1^i, \dots, q_4^i instead of two, and are still connected to cycles via $\{p_1^i, c_1^i\}$ resp. $\{q_1^i, d_1^i\}$. Edges $\{p_2^i, d_1^i\}, \{p_2^i, d_3^i\}$ and $\{q_2^i, c_1^{i+1}\}, \{q_2^i, c_3^{i+1}\}$, which connect paths with nodes of the “next” cycle, are replaced by $\{p_4^i, q_2^i\}, \{p_4^i, d_3^i\}$, resp. $\{q_4^i, p_2^{i+1}\}, \{q_4^i, c_3^{i+1}\}$. During the game adversary B does not insert any additional edges. All nodes have degrees two or three. In particular, nodes p_2^1, p_3^1 have degree three resp. two: in the first round B presents the highest priority data item $\langle u, d_u, b_u, v \rangle$

and relabels nodes such that A picks edge $\{p_2^1, p_3^1\}$, for any degree $d_u \in \{2, 3\}$ and any bit $b_u \in \{0, 1\}$. The remainder of the left cycle and path in T_1 now forms a fragment of the graph (see gray edges in the figure). Therein A can pick at most two edges: algorithm A scores three out of four edges for the left cycle and path of T_1 . Analogously, adversary B repeats this strategy for the remaining paths and cycles containing edges $\{q_2^1, q_3^1\}$, $\{p_2^2, p_3^2\}$, $\{q_2^2, q_3^2\}$, \dots , $\{q_2^k, q_3^k\}$. Hence we obtain the claimed convergence to $\frac{3}{4} = \frac{\Delta}{2\Delta-2}$.

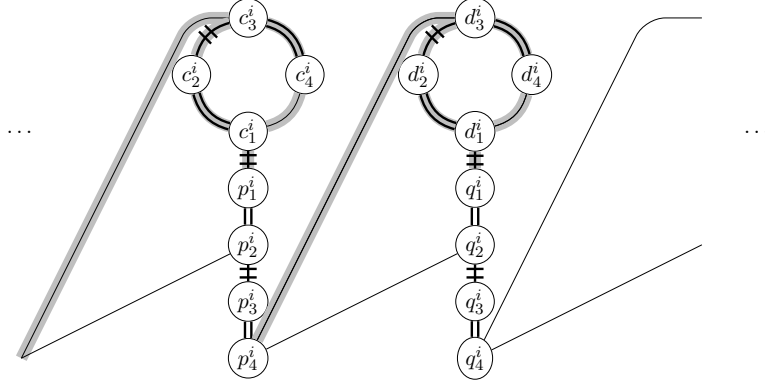


Figure 31: The i -th trap T_i for $\Delta = 3$ (gray edges form fragments of the graph during the game)

Perfect Matching. Similar to the cycle on nodes e_1, \dots, e_4 , adversary B replaces the extra node e_0 with a length-four cycle C . Nodes c_1^1, c_3^1 of the first cycle (resp. nodes p_2^1, c_3^1 for $\Delta = 3$) are connected to different nodes of C such that degrees in C are two and three and the graph is bipartite. As discussed above, the construction starts with the left cycle and path of trap T_1 . However, when the star centered at node c_3^1 is disconnected from the rest of the traps, node c_3^1 is still connected with the nodes in C . W.l.o.g. we assume that A instantly isolates all nodes in the star and in C . (To compensate for the two additional edges scored by A in C , adversary B increases k accordingly.) Thereafter the construction proceeds as shown above. \square

Note. Theorem 62 carries over to graphs in which the average degree is at most Δ . However, bounding the average degree gives an adversary more freedom to construct a hard instance, since the maximum degree may be larger than Δ . Therefore stronger inapproximability bounds might apply. Indeed, even if the average degree is a *constant* then all \mathcal{QC} -algorithms, the “one-sided” algorithms KARPSSIPER, 1-2-GREEDY, MINGREEDY, and STATIC-MINGREEDY, as well as the “double-sided” algorithms MDS and DOUBLE-MINGREEDY do not beat factor $\frac{1}{2}$. Why? Similar to the construction in the proof of

Theorem 62, consider the graph which consists of only one trap, call it T_1 , and observe that no edges have to be added to make an algorithm pick edges $\{x_1^1, y_1^1\}, \dots, \{x_\Lambda^1, y_\Lambda^1\}$ in the first $\Lambda = \Theta(\Delta)$ rounds. Thereafter, only a constant number of additional edges can be picked, hence the approximation ratio converges to $\frac{1}{2}$ for $\Delta \rightarrow \infty$. Only four nodes have degree Δ , namely the cycle nodes c_1^1, c_3^1, d_1^1 , and d_3^1 . All other $\Theta(\Delta)$ many nodes have constant degree. Consequently, for $\Delta \rightarrow \infty$ the average degree is constant.

Note. We compare the class \mathcal{DS}_{01} with the class of *degree-based* adaptive priority algorithms [BBLM10]. In each round a degree-based algorithm submits a priority order of degrees, i.e. an order of positive integers, and receives a node whose degree has highest priority (the algorithm additionally receives the neighbors of a node in the input graph, similar as in the definition of the class \mathcal{VA}).

Poloczek [Pol12] gave a construction showing that degree-based adaptive priority algorithms fail with approximation ratio $\frac{1}{2}$ on bipartite graphs. However, degree based algorithms request nodes w.r.t. their degree in the input graph, i.e. an algorithm cannot request *current* degrees. Therefore the result does not apply to e.g. MINGREEDY or the KARPSSIPER algorithm, as does Theorem 62.

Node Adjacencies. We proceed with results on algorithms capable to detect node adjacencies. In particular, we study \mathcal{VA} -algorithms for general graphs (Lemma 63 and Theorem 64) and bipartite graphs (Theorem 65). For general graphs, the proof is arranged in two parts. Recall that the set of nodes is given to the algorithm as a priori knowledge. However, an adversary committing to a fixed number of nodes has limited flexibility to react to the moves of the algorithm as opposed to a variable number of nodes. Therefore, as a first step we show in Lemma 63 that factor $\frac{\Delta-1}{2\Delta-3}$ cannot be beaten if the set of nodes is *not* known in advance. Thereafter, in Theorem 64 we show how to adapt the construction to a priori knowledge on the full set of nodes.

Lemma 63. *Let $A \in \mathcal{VA}$, and assume that A does not receive the set of nodes in advance. There is an input graph of degree at most Δ for which A computes a matching of size at most $\frac{\Delta-1}{2\Delta-3}$ times optimal.*

Proof. We describe the adaptive priority game against an adversary B . The game consists of the *regular game*, which lasts for the first $s = \Delta - 3$ rounds, followed by the *endgame*, which has two rounds and ends with the last round.

During the regular game, adversary B maintains the following invariant: each node v that is not yet isolated has a data item of one of the following types.

Type 1: In data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ nodes u and v_1, \dots, v_{D_u} are *unknown*, i.e. they did not occur in a previously received data item, and $3 \leq D_u \leq \Delta$ holds.

Type 2: In data item $\langle u, v_1, v_2, v \rangle$ nodes u and v_1, v_2 are unknown.

Type 3: In data item $\langle u, v_1, v_2, v_3, v \rangle$ nodes u and v_1, v_2 are unknown and v_3 is *known*, i.e. node v_3 was received by A in a data item of a previous round.

In particular, all nodes in G have degree at least two, i.e. the set of *allowed* degrees is $D = \{2, \dots, \Delta\}$, cf. Section 5.3.1.

Consider the very first round. Since all nodes are still unknown, all nodes have data items of type 1 or 2. Hence the invariant holds. Consider round i and assume that the invariant holds. Adversary B presents the highest ranked data item that is of type 1, 2, or 3. Call that data item a_i .

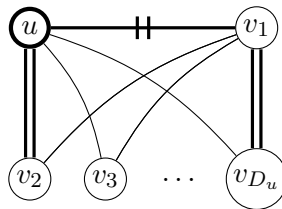


Figure 32: A connected component of a hard instance (the algorithm receives a type-1 data item for the bold node u)

Case 1: $a_i = \langle u, v_1, \dots, v_{D_u}, v \rangle$ is a type-1 data item. Since all nodes in a_i are unknown, we may w.l.o.g. assume that A matches u with $v = v_1$. Adversary B constructs the connected component C depicted in Figure 32 which consists only of nodes of types 1 and 2, since all nodes in C are unknown and have degree at least two. All nodes of C are isolated in the next round, hence the invariant is maintained. Observe that within component C the maximum matching scores two edges (the double edges $\{u, v_2\}, \{v_1, v_{D_u}\}$ in Figure 32) whereas A scores only one edge (the crossed edge $\{u, v_1\}$).

Since in Case 1 algorithm A requests only unknown nodes, adversary B is able to trick A into unveiling part of G from which A cannot gather knowledge about the rest of G , namely component C .

Can A act smarter? Assume that A has already received the data items of the *middle nodes* m_1, \dots, m_k of the triangles $\{l_j, m_j, r_j\}$ of known nodes connected by *frontier nodes* r_j and unknown nodes u_j to the still unknown *center* of G , see Figure 33. If A

requests an unknown node with two unknown neighbors, then B tricks A by constructing a new triangle $\{l_i, m_i, r_i\}$.

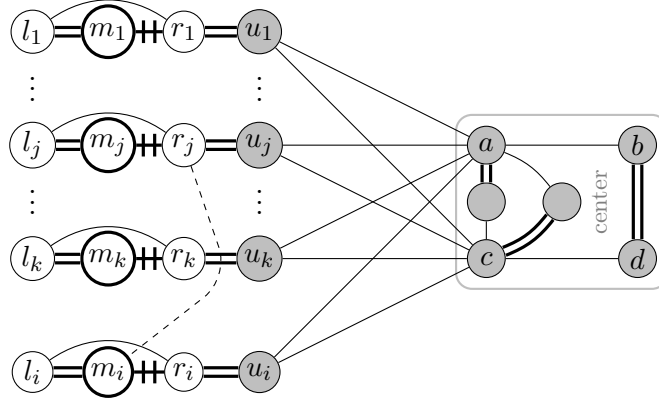


Figure 33: The center of a hard instance (gray nodes are unknown, the algorithm receives data items for bold nodes, the dashed edge is an example for $\{m_i = v, r_j = v_3\}$ in case 3)

Case 2: $a_i = \langle u, v_1, v_2, v \rangle$ is a type-2 data item. Again, all nodes of a_i are unknown and we may assume that A matches u with $v = v_1$. Adversary B constructs a triangle $\{l_i, m_i, r_i\}$ with $l_i = v_2, m_i = u, r_i = v_1$ and inserts edge $\{r_i, u_i\}$, with a new unknown node u_i , to connect the triangle to the unknown center. Observe that before nodes m_i, r_i are matched, nodes m_i, l_i are of type 2 and r_i, u_i are of type 1. After matching m_i, r_i , nodes l_i, m_i, r_i are isolated and u_i turns into a type-3 node. Hence the invariant still holds. Again, a maximum matching scores two edges, namely $\{l_i, m_i\}, \{r_i, u_i\}$, and A scores the edge $\{m_i, r_i\}$.

Now assume that A tries to explore the neighborhood of known nodes. Observe that the only data items with a known node are of type 3 and have exactly one unknown node: since the known nodes l_j, m_j, r_j are already isolated, an unknown node can only be explored in the neighborhood of frontier nodes. Again, adversary B tricks A with a new triangle $\{l_i, m_i, r_i\}$.

Case 3: $a_i = \langle u, v_1, v_2, v_3, v \rangle$ is a type-3 data item. Since v_3 is known, v_3 occurred in a previously presented data item. Observe that in our construction so far, the only type-3 nodes are unknown neighbors of known frontier nodes. So u is the neighbor of a frontier node $r_j = v_3$ with $j < i$.

Is algorithm A successful in exploring the unknown neighbor u_j of r_j , i.e. does $u = u_j$ hold? Not necessarily, since B may on the fly construct further neighbors of r_j .

Why? Since r_j gets matched as soon as it becomes known, algorithm A never gets to see the data item of r_j and consequently A can never tell if it already knows all neighbors of r_j . (Adversary B uses this trick here as well as in the end game.)

Since $v_3 = r_j$ is matched and v_1, v_2 are unknown, we may assume that A matches u with $v = v_1$. Adversary B behaves exactly as in case 2 and constructs the triangle $\{l_i = v_2, m_i = u, r_i = v_1\}$ and inserts the edge $\{r_i, u_i\}$ where u_i is a new unknown node. To complete its trick, adversary B also inserts the edge $\{m_i = u, r_j = v_3\}$ (see e.g. the dashed edge in Figure 33). Before m_i, r_i are matched, node l_i is of type 2, nodes r_i, u_i are of type 1 and $m_i = u$ is of type 3. After matching m_i, r_i , nodes l_i, m_i, r_i are isolated and u_i turns into a type-3 node. Node u_j is still of type 3. Hence the invariant still holds. As in case 2, a maximum matching scores edges $\{l_i, m_i\}, \{r_i, u_i\}$ whereas A scores $\{m_i, r_i\}$.

This concludes the regular game. We show later that in the first round of the endgame adversary B makes algorithm A match the center nodes a and b . Hence in the next and last round algorithm A matches node c . So algorithm A scores two edges in the center, whereas three edges are optimal. As claimed, we get

$$|M| = s + 2 = \Delta - 1 \quad \text{and} \quad |M^*| = 2s + 3 = 2\Delta - 3,$$

where M^* and M denote a maximum matching resp. the matching computed by A .

Observe that our invariant still holds in the first round of the endgame. Again, adversary B presents the highest ranked data item of type 1, 2, or 3. Nodes a and c are the only type-1 nodes left, since nodes u_j have known neighbors and are of type 3 and all other nodes have degree two and are of type 2. The degree of a and c is $\delta \leq 3 + s$, since both a and c have three center neighbors and in each round of the regular game at most one neighbor is added to a resp. c . Let $a_{\Delta-2}$ be the data item received in round $s + 1 = \Delta - 2$.

Case 4a: $a_{\Delta-2} = \langle u, v_1, \dots, v_\delta, v \rangle$ is a type-1 data item. Since all nodes of $a_{\Delta-2}$ are unknown we may assume that A matches u with $v = v_1$. Adversary B sets $u = a$, $v_1 = b$ and v_2, \dots, v_δ as the remaining neighbors of a .

Case 4b: $a_{\Delta-2} = \langle u, v_1, v_2, v \rangle$ is a type-2 data item. Since all nodes of $a_{\Delta-2}$ are unknown we may assume that A matches u with $v = v_1$. Adversary B sets $u = b$, $v_1 = a$ and $v_2 = d$.

Case 4c: $a_{\Delta-2} = \langle u, v_1, v_2, v_3, v \rangle$ is a type-3 data item. As in case 3, the known node v_3 is some matched frontier node $r_j, j < \Delta - 2$ and we may assume that A matches u with $v = v_1$, since v_1, v_2 are unknown. As in case 3, adversary B does *not* present the data item of the unknown node u_j . Instead, B makes b a neighbor of r_j by inserting the edge $\{v_3=r_j, b\}$ —now b has three neighbors—and sets $u=b, v_1=a$, and $v_2=d$.

Adversary B does not violate degree constraints. Each node u introduced in case 1 has degree at most $D_u \leq \Delta$. All other degrees are at most three, but for nodes a and c and for frontier nodes r_j . As discussed, nodes a and c have degree at most $\delta = 3 + s \leq \Delta$. Frontier node have degree at most $3 + (s - 1) + 1 = \Delta$, since in each but the first round of the regular game and in round $s + 1$ at most one incident edge is added. \square

Theorem 64. *Let $A \in \mathcal{VA}$. For any $\varepsilon > 0$, there is an input graph of degree at most Δ for which A computes a matching of size at most $\frac{\Delta-1}{2\Delta-3} + \varepsilon$ times optimal.*

Proof. We modify the adversary who constructs hard inputs in the proof of Lemma 63. The basic idea is to construct a large number connected components, each of which is a graphs like in Lemma 63 with a variable number of nodes and “local” approximation ratio at most $\frac{\Delta-1}{2\Delta-3}$. Thereafter, to commit to the announced number of nodes, the adversary constructs a small portion of the graph which may be solved optimally by the algorithm. The number of edges scored by the algorithm in this portion is only constant. Thus the overall approximation ratio converges to $\frac{\Delta-1}{2\Delta-3}$ for a large number of nodes.

Before the first round, the modified adversary B' announces the number $t \cdot \Delta^2$ of nodes, where $t \geq 1$ is integer. Throughout the proof we frequently refer to the data item types and cases found in the proof of Lemma 63 (types 1, 2, and 3 and cases 1, 2, 3, and 4a-c). Again, the game between A and B' is split up into the regular game and the endgame. In each round of the regular game, again, the construction of B' keeps up the invariant that all non-isolated nodes in the graph have data items of type 1, 2, or 3, and, again, adversary B' returns the highest priority data item having one of these types.

However, depending on the requests of A , not only one but several centers C_1, C_2, \dots might be constructed, each in its own connected component of G . Each center C_i is defined as in the proof of Lemma 63, with nodes a_i, b_i, c_i , and d_i and two more nodes unique to C_i , see Figure 33. Each C_i gets attached to it some triangles which are not connected to any other center. Thereafter, the nodes of C_i are matched in the same order as in the proof of Lemma 63, i.e., when no more triangles are attached, nodes a_i and b_i are matched to each other before c_i is matched. Once c_i is matched, all edges in the connected component of C_i are removed.

As we shall prove, the large number of nodes enables B' to construct a large number of connected components, each of which is like the graph constructed in Lemma 63 with approximation ratio exactly $\frac{\Delta-1}{2\Delta-3}$. A negligible portion of the graph may be solved optimally by A . This does not increase the overall approximation ratio, asymptotically.

Assume that B' has already created the centers C_1, \dots, C_l . Call C_i *active* if a_i is not yet matched with b_i , and *inactive* otherwise. The construction asserts that C_1, \dots, C_{l-1} are inactive; C_l might still be active. (We note here that after center C_l becomes inactive, nodes in the *fragment* (cf. Section 5.3.1) consisting of the star centered at node c_l do not have data items of types 1, 2, or 3. However, as in Lemma 63 we may assume that algorithm A scores an optimal number, namely one, of edges in this fragment. Therefore, the additional data item types do not have effect on the rest of the construction and we do not discuss these types explicitly.)

- Assume that in the next round A receives a **type-2** data item of a node with two unknown neighbors. Assume that C_l is already inactive, then B' creates the next center C_{l+1} and connects a new type-2 triangle to a_{l+1}, c_{l+1} as described in case 2.

If otherwise C_l is still active, let δ be the number of neighbors of a_l, c_l constructed so far and recall that we demand $\delta \leq \Delta$. If $\delta < \Delta$ holds, then B' connects a new type-2 triangle to a_l, c_l as described in case 2. If $\delta = \Delta$, then B' makes A match a_l with b_l as described in case 4b, thereby inactivating C_l .

- Assume that in the next round A receives a **type-3** data item. By construction, the received node is unknown and among its three neighbors there is exactly one known node v_3 , where $v_3 = r$ is a frontier node r in the connected component of the still active center C_l . Let δ be the number of neighbors of a_l, c_l constructed so far. If $\delta < \Delta$, then B' connects a new type-3 triangle to a_l, c_l as described in case 3. If $\delta = \Delta$, then B' makes A match a_l with b_l as described in case 4c, and inactivates C_l .

- Assume that in the next round A receives a **type-1** data item. If the degree of the received node is smaller than Δ , then B' proceeds as in case 1 and creates a new type-1 connected component.

Now assume that the received node has degree Δ . If nodes a_l, c_l have degree less than Δ , then again B' creates a new type-1 connected component. If nodes a_l, c_l have degree $\delta = \Delta$, then B' makes A match a_l with b_l as described in case 4a, and thereby inactivates C_l .

Why is C_l inactivated before B' constructs the next active center? In type-2 and type-3 rounds an increasing number of triangles is connected to C_l until the degrees of a_l, c_l are Δ . (In intermediate type-1 rounds only type-1 connected components are constructed.) Thereafter, C_l is inactivated in the first type-2 or type-3 round or the first type-1 round in which a data item of a degree- Δ node is received. (In intermediate type-1 rounds with nodes of degree less than Δ only type-1 connected components are constructed.)

The endgame begins as soon as B' has constructed $k \geq t \cdot \Delta^2 - 6\Delta$ nodes. Let $\nu = t \cdot \Delta^2 - k$ be the number of nodes still to be constructed. We have $6\Delta \geq \nu \geq 2\Delta$, since in each round no more than 4Δ nodes are introduced (e.g. when $\Delta = 3$ holds and a new triangle is connected to a new center).

Since B' has committed to a number of exactly $t \cdot \Delta^2$ nodes, in the first round of the endgame B' utilizes all remaining ν nodes to create additional connected components $\Gamma_1, \dots, \Gamma_c$, each a complete bipartite graph with $n_r = 2$ nodes on the right side and $2 \leq n_l \leq \Delta$ nodes on the left. Adversary B' constructs large left sides such that $c = O(1)$ is constant and all nodes have degree at least two. All nodes in $\Gamma_1, \dots, \Gamma_c$ are still unknown, in particular all nodes have data items only of types 1 or 2. Since $n_r = 2$, each Γ_i has at most two edges in a maximum matching, making an additional constant number $2c$ of optimal edges in total. We assume that A performs optimally in all Γ_i , thereby scoring $2c$ edges.

Also during the endgame, A matches still unmatched nodes in the already inactive centers C_1, \dots, C_{l-1} , if they were not matched during the regular game. Moreover, we assume that A performs optimally also in the connected component of the last center C_l .

We bound the approximation ratio of A . Recall that A scores only one out of two edges in each type-1 connected component, therefore to bound the performance of A we may assume that no type-1 components were constructed. Since each center C_i and each triangle has a constant number of nodes and C_i is connected with at most $\Delta - 3$ triangles, the connected component of C_i has $O(\Delta)$ nodes. The inactive centers C_1, \dots, C_{l-1} each have a maximum number of triangles, hence their components have $\Theta(\Delta)$ nodes each. Since only C_l might be active at the end of the regular game, there are at least

$$l - 1 = \Omega(t\Delta)$$

centers being inactivated. But A scores $\Delta - 1$ out of $2\Delta - 3$ edges in the component of an inactive center, hence the approximation ratio of A is at most

$$\frac{(l-1) \cdot (\Delta-1) + (2\Delta-3) + 2c}{(l-1) \cdot (2\Delta-3) + (2\Delta-3) + 2c},$$

since A performs optimally in C_l and in $\Gamma_1, \dots, \Gamma_c$. Letting $t \rightarrow \infty$ we get $l \rightarrow \infty$ and this ratio is dominated by $\frac{(l-1) \cdot (\Delta-1)}{(l-1) \cdot (2\Delta-3)}$. The statement follows. \square

Next, we adapt the construction used for $\mathcal{W}\mathcal{A}$ -algorithms to bipartite graphs.

Theorem 65. *Let $A \in \mathcal{W}\mathcal{A}$. For any $\varepsilon > 0$, there is a bipartite input graph for which A computes a matching of size at most $\frac{3}{4} + \varepsilon$ times optimal.*

Proof. We describe the adaptive priority game between A and an adversary B who proceeds very similar to the adversary in the proofs of Theorem 64 and Lemma 63. The set of $\Delta^2 + 4$ nodes is known in advance to algorithm A , and all degrees are bounded by at most Δ . Here, the *center* of the graph has four nodes. We reuse the notation from the proof of Lemma 63: in particular, we use the same types of data items.

In each round during the *regular game* adversary B keeps up the invariant that all *known* nodes are matched (recall that a node is called *known* as soon as A has received its name in a data item). Adversary B presents the highest priority data item which has one of the following types.

Case 1: In a *type-1* data item $\langle u, v_1, \dots, v_{D_u}, v \rangle$ all nodes are *unknown* and u has degree $3 \leq D_u \leq \Delta$. W.l.o.g. we may assume that A matches u with $v = v_1$.

Adversary B constructs an extra connected component which is not connected to the center. Since the graph is bipartite, the extra component may not contain triangles and hence B cannot proceed as in Lemma 63. Instead, adversary B constructs the extra connected component depicted in Figure 34.

Observe that the extra component contains four edges in a maximum matching (the double edges), and that besides edge $\{u, v_1\}$ algorithm A can pick at most two more edges in the extra component. W.l.o.g. we assume that A picks two more edges *instantly*, i.e. these rounds of the game do not need to be analyzed explicitly. Thereafter all known nodes are matched.

Case 2: In a *type-2* data item $\langle u, v_1, v_2, v \rangle$ all nodes are *unknown*. W.l.o.g. we may assume that A matches u with $v = v_1$.

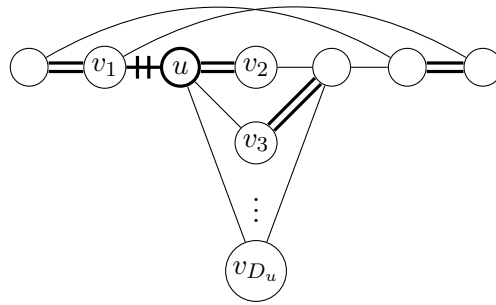


Figure 34: An extra component (the algorithm receives a type-1 data item for the bold node u)

Adversary B constructs part of the graph which is connected to the center. However, since the graph must be bipartite, adversary B may not construct a *triangle* as in Lemma 63, but rather constructs the *gadget* in Figure 35. Nodes are relabeled such that we have $u = m_i$, $v = r_i$, and $v_2 = l_i$.

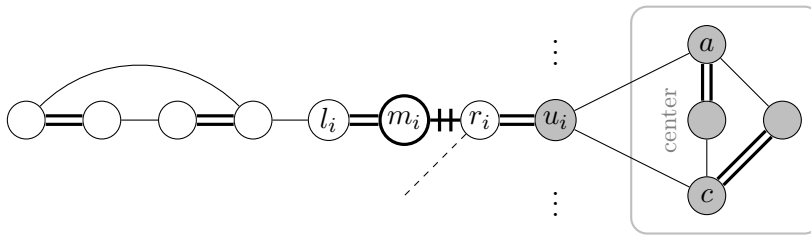


Figure 35: A gadget connected to the center (the dashed edge illustrates case 3)

Again, algorithm A instantly scores two more edges in the gadget, whereas the gadget contains four edges in a maximum matching. Node u_i remains unknown throughout the rest of the regular game, as follows from the next case.

Case 3: In a *type-3* data item $\langle u, v_1, v_2, v_3, v \rangle$ nodes u, v_1 , and v_2 are *unknown* and v_3 is known. Since known nodes are already matched, w.l.o.g. we may assume that A matches u with $v = v_1$.

Nodes are relabeled such that we have $u = m_i$, $v = r_i$, and $v_2 = l_i$. By construction, node v_3 is a frontier node r_j of some previously constructed gadget, i.e. we have $j < i$, since frontier nodes are the only known nodes with neighbors which are not yet matched. Adversary B adds the required edge to the construction (the dashed edge in Figure 35 illustrates such an additional edge): this is possible, since algorithm A never gets to know the complete set of neighbors of r_j .

Again, we assume that A instantly scores two more edges in the gadget. As in case 2, node u_i remains unknown.

Since each extra component and each gadget contains at most $\Delta + 6$ nodes and the graph contains $\Delta^2 + 4$ nodes in total, the above construction can be repeated at least $k = \Omega(\Delta)$ times. Hence the approximation ratio of A is bounded by at most

$$\lim_{\Delta \rightarrow \infty} \frac{3k + O(1)}{4k + O(1)} = \frac{3}{4},$$

where constants account for the following edges scored by A .

- A constant number of edges incident with center nodes a and c , illustrated by double drawn edges in Figure 35.
- Edges connecting nodes which B introduces in the *endgame* to deliver on its promise that the graph contains $\Delta^2 + 4$ nodes: once too few nodes are left to construct another gadget or extra component, adversary B puts the remaining nodes in their own connected component. This component is complete bipartite with exactly two nodes in one of the partitions: algorithm A scores two edges.

For consistency, adversary B enforces a minimum degree of two for all nodes in the extra component by including sufficiently many nodes. Therefore B asserts that adding the last gadget or extra component does not leave too few nodes, namely by limiting the size of the last extra component accordingly (the size of a gadget is always limited by a constant). \square

Note. The inapproximability bound in Theorem 65 also holds for algorithms which may choose the partition of the nodes. Therefore the construction is changed slightly such that the graph contains two centers instead of one, and each gadget is connected to one of the centers depending on which partition the node u matched first in the gadget belongs to.

“Double-Sided” Algorithms. The MDS algorithm does not perform better than the KARPSSIPER algorithm, as we show next. Thereafter we show that also the DOUBLE-MINGREEDY algorithm as well as all other \mathcal{DSE}_{01} -algorithms perform no better than KARPSSIPER if degrees are bounded by $\Delta = 3$.

Theorem 66. *For each $\Delta \geq 3$ there is a bipartite graph of degree at most Δ for which MDS computes a matching of size at most $\frac{\Delta}{2\Delta-2}$ times optimal.*

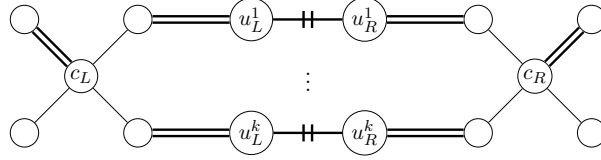


Figure 36: A hard instance for MDS

Proof. Choose $k = \Delta - 2$. The hard instance is depicted in Figure 36. Observe that the degree sum of any edge is at least 4, since nodes c_L and c_R have degree at least 3. Hence we may assume that in the first step MDS picks edge $\{u_L^1, u_R^1\}$ (the top crossed edge). Assume that edges $\{u_L^1, u_R^1\}, \dots, \{u_L^i, u_R^i\}$ with $i < k$ have already been picked. The minimum degree sum is still four, and edge $\{u_L^{i+1}, u_R^{i+1}\}$ is picked next. In the end, for each of nodes c_L and c_R an incident edge is picked.

Hence the computed matching has $k + 2 = \Delta$ edges, whereas a maximum matching consists of the $2k + 2 = \Delta - 2$ double drawn edges. \square

Theorem 67. *Let $A \in \mathcal{DSE}_{01}$. For any $\varepsilon > 0$, there is a bipartite input graph of degree at most $\Delta = 3$ for which A computes a matching of size at most $\frac{3}{4} + \varepsilon = \frac{\Delta}{2\Delta-2} + \varepsilon$ times optimal.*

Proof. We slightly change the adversary B from the proof of Theorem 62 to obtain an adversary B' for A . Adversary B' removes right paths and cycles and their incident edges from all traps, and connects the path node p_4^i to nodes c_3^{i+1}, p_2^{i+1} of the next cycle and path. Cycle C and the cycle on nodes e_1, e_2, e_3, e_4 along with all their incident edges are replaced by two edges in the first and the k -th cycle and path, namely edges $\{c_3^1, p_1^1\}, \{c_2^1, p_2^1\}$ resp. $\{c_4^k, p_4^k\}, \{p_1^k, p_4^k\}$, see Figure 37. The length k of the chain is determined by B' during the game based on the actions taken by A .

Before the first round, adversary B' announces that the number of nodes is $8n$, for some large integer n . Besides the chain of paths and cycles, the graph has $n - k$ additional connected components, each with two length-four cycles connected by two edges like in Figure 38. Observe that any edge in any connected component is incident either with a degree-2 node and a degree-3 node or with two degree-3 nodes. No edge is incident with two degree-2 nodes.

The following INVARIANT holds at the beginning of each round $3i + 1$ (for $i \geq 0$) throughout the adaptive priority game: so far, the chain contains $0 \leq k^* \leq i$ paths and

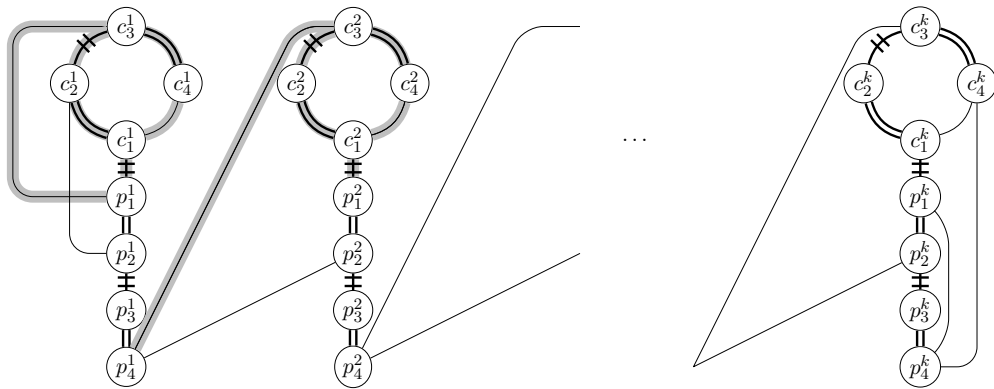


Figure 37: The construction of adversary B' : no additional edges are inserted during the game

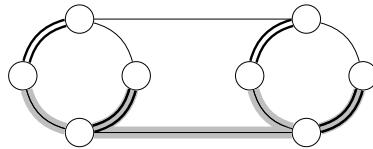


Figure 38: An additional component

cycles and there are $i - k^*$ additional connected components. In particular, all nodes in these parts of the graph are isolated, but node $p_4^{k^*}$ of the currently “rightmost” path (if any). Observe for $i = 0$: in the first round no nodes are isolated and the INVARIANT holds.

Consider round $3i + 1$. Observe that the minimum degree is two (adversary B does not allow degree 1, cf. Section 5.3.1), and that every edge is incident with at least one node of degree three. Adversary B' presents the highest priority data item $\langle (u, d_u, b_u), (v, d_v, b_v) \rangle$ in the order submitted by A with $d_u, d_v \in \{2, 3\}$ and at least one of d_u, d_v equals three. The original degree bits may take on any value, since the construction of B asserts that for each received data item the current and original degrees of both nodes are the same.

If $d_u = d_v = 3$, then B' constructs the next additional component and relabels nodes such that u and v are the two topmost nodes in Figure 38. Observe that A scores at most three out of four edges in this component, since in the remaining *fragment* (cf. Section 5.3.1) only gray edges are left. W.l.o.g. we assume that A scores the additional two edges in the next two rounds. Since k^* is not increased, the INVARIANT continues to hold. Here, current and original degrees of the received nodes might in fact be different, however, since this part of the graph is disconnected from the rest algorithm A cannot take any advantage.

If $d_u \neq d_v$, then w.l.o.g. let $d_u = 3$ and $d_v = 2$. By the INVARIANT, algorithm A has already matched nodes $p_2^1, p_3^1, p_2^2, p_3^2, \dots, p_2^{k^*}, p_3^{k^*}$ in previous rounds. Adversary B' relabels nodes such that $u=p_2^{k^*+1}$ and $v=p_3^{k^*+1}$ hold. After $p_2^{k^*+1}$ and $p_3^{k^*+1}$ are matched, the remainder of the k^*+1 -th path and cycle forms a fragment of the graph, see gray edges in Figure 37. In this fragment A scores at most two more edges. W.l.o.g. we assume that A does so in the next two rounds. (Again, algorithm A cannot take advantage of differing current and original degrees.) Hence k^* is incremented by one and the INVARIANT holds before round $3(i+1)+1$.

We assume that the last path and cycle resp. the last additional component is solved optimally, i.e. algorithm A scores four out of four edges. In each other path and cycle and in each other additional component, algorithm A scores three out of four edges. Hence B' can choose sufficiently large n such that the approximation ratio of A is at most $\frac{(n-1) \cdot 3 + 4}{4} \leq \frac{3}{4} + \varepsilon$. \square

For bipartite graphs, “double-sided” algorithms in class \mathcal{DSE}_{01} (like e.g. DOUBLE-MINGREEDY) perform only scarcely better (if at all) than \mathcal{DS}_{01} -algorithms, as we show next.

Theorem 68. *Let $A \in \mathcal{DSE}_{01}$. There is a bipartite input graph of degree at most $\Delta \geq 3$ for which A computes a matching of size at most $\frac{\Delta+1}{2\Delta-2}$ times optimal.*

Proof. The adaptive priority game between A and an adversary B lasts for $\Delta+1$ rounds. Let $\delta = \Delta - 3$. The final construction G contains the graph G' depicted in Figure 39 as a subgraph. In particular, graph G contains additional edges which are not depicted in G' , but G does not have any additional nodes.

The construction of B proceeds such that in rounds $1, \dots, \delta$ algorithm A picks edges $\{u_1, v_1\}, \dots, \{u_\delta, v_\delta\}$ and after round δ the reduced graph consists only of gray nodes and of edges connecting gray nodes. Observe that all remaining edges touch exactly four gray nodes, namely the unlabeled ones in the figure. We assume that in this reduced graph algorithm A scores four edges in four rounds, which is optimal. Since G' contains a perfect matching of size $2\Delta - 2$ and A scores one edge in each of $\delta + 4 = \Delta + 1$ rounds, the approximation ratio is $\frac{\Delta+1}{2\Delta-2}$, as claimed. In the rest of the proof it remains to discuss the first δ rounds.

Recall that A does not receive identifiers of neighbors of the nodes in a data item. As a consequence, in each round adversary B is free—without being inconsistent—to relabel nodes in G' according to the data item presented to A .

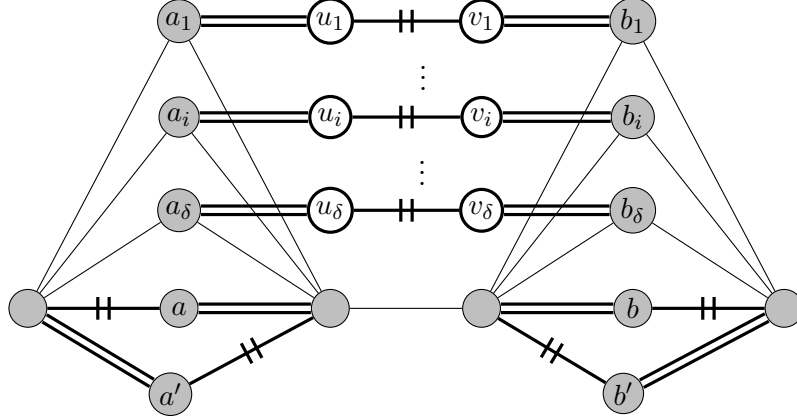


Figure 39: The subgraph G' of the final construction (the algorithm receives data items for bold nodes)

We proceed inductively. Assume that A has picked edges $\{u_1, v_1\}, \dots, \{u_{i-1}, v_{i-1}\}$ and consider round i . The minimum degree in the reduced graph G_i is 2. Adversary B uses the set $D = \{2, \dots, \Delta\}$ of *allowed* degrees, cf. Section 5.3.1. From the order submitted by A in round i adversary B presents the highest priority data item $\langle (u, d_u, b_u), (v, d_v, b_v) \rangle$ with $d_u \in D$ and $d_v \in D$. Bits b_u and b_v are ignored: the construction asserts that the degrees of nodes u and v have not changed since the beginning of the game, as we show below. Adversary B relabels nodes such that $u = u_i$ and $v = v_i$ hold: algorithm A picks edge $\{u_i, v_i\}$, as desired.

Now B delivers on its promise that both nodes have degree d_u resp. d_v . Therefore B inserts additional edges into the graph. In particular, since u_i already has two incident edges in G' , adversary B adds $d_u - 2$ edges, each incident with u_i and one of nodes $a, a', a_1, \dots, a_\delta$. Analogously, adversary B adds $d_v - 2$ edges incident with v_i and one of nodes $b, b', b_1, \dots, b_\delta$.

It remains to show that B does not violate degree constraints when inserting new edges. Since we have $d_u \leq \Delta$ and thus $d_u - 2 \leq \Delta - 2$, for all u -nodes at most $\delta(\Delta - 2) = (\Delta - 3)(\Delta - 2) = (\Delta - 3)^2 + (\Delta - 3)$ edges are inserted. Since all a -nodes can receive up to $\delta(\Delta - 3) + 2(\Delta - 2) = (\Delta - 3)^2 + 2(\Delta - 2)$ edges, their degrees are increased to at most Δ if new edges are distributed evenly. Analogously, degrees of b -nodes are at most Δ . \square

Note. This construction also allows to show the same bound if an algorithm may additionally choose the partition of each node.

Degree Sensitivity for Paths. We continue with a result on algorithms picking entire paths instead of single edges.

Theorem 69. *Let A be a \mathcal{DSP}_{01} -algorithm. For each $\varepsilon > 0$ there is a bipartite graph on which A computes a matching of size at most $\frac{1}{2} + \varepsilon$ times optimal.*

Proof. We present the adaptive priority game between A and an adversary B . The family of graphs constructed by B will be such that the approximation ratio achieved by A degrades as the maximum degree Δ is increased.

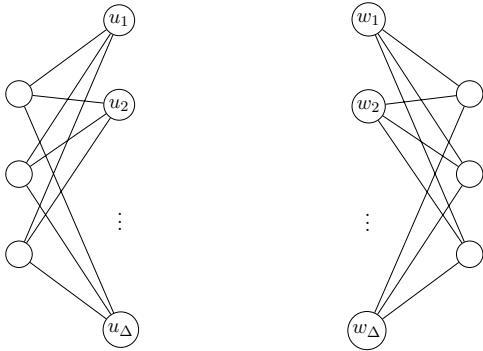


Figure 40: Hard input instances for \mathcal{DSP}_{01} -algorithms: the subgraph G'

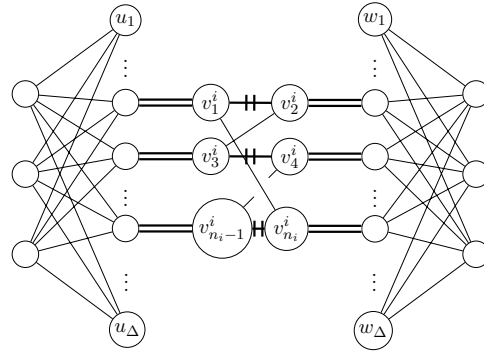


Figure 41: A path P^i is added (optimal edges are drawn double, edges picked by A are crossed, deferred edges connecting path nodes with u -nodes resp. w -nodes are not drawn)

The final construction has $4\Delta + 6$ nodes and contains the subgraph G' of $2\Delta + 6$ nodes depicted in Figure 40 (node labels given there are not actual node names, but rather used to discuss the construction). Further 2Δ nodes and incident edges are added to G' depending on the actions taken by A . In particular, further Δ nodes are added to each partition.

We first give an overview over the game and argue the claimed approximation ratio. Thereafter we discuss the construction in detail.

During the *regular game*, which begins with the first round, data items received by A contain solely newly added nodes which are not contained in G' (this can be accomplished by relabeling nodes accordingly). The same number of nodes is added to the left as well as to the right side of the bipartition, where at most ℓ nodes are added to each partition per round since A may pick at most ℓ edges at a time. Denote by ℓ_i the total number of nodes added to each side after round i (we have $\ell_0 = 0$). The regular game ends after the first round i for which $\Delta - \ell - 1 \leq \ell_i \leq (\Delta - \ell - 1) + (\ell - 1)$ holds.

Next, we discuss additional edges. The final construction asserts that *each* of the nodes matched by A during the regular game is incident with a *distinct* optimal edge in a maximum matching. Hence A rules out twice as many optimal edges as it picks. In favor of A , we assume that only $\Delta - \ell - 1$ edges are picked during the regular game and thereafter A scores an optimal number in the reduced graph.

The *end game* starts when the regular game is over. Adversary B adds r nodes to each partition, where $\ell + 1 \geq r \geq 2$ is chosen such that Δ nodes are added to each partition in total. More edges are added only incident with these new $2r$ nodes. We assume that in the reduced graph A computes an optimal matching, which has at most $2\ell + 2 + 6$ edges. Why? At most $2r \leq 2\ell + 2$ nodes are added, all new edges are incident with the newly added nodes, and all other edges (which are contained in G') are incident with the three leftmost nodes and the three rightmost nodes in Figure 40.

Consequently, since ℓ is a constant and hence $\lim_{\Delta \rightarrow \infty} \frac{\ell}{\Delta} = 0$ holds, algorithm A achieves approximation ratio at most

$$\lim_{\Delta \rightarrow \infty} \frac{(\Delta - \ell - 1) + (2\ell + 8)}{2(\Delta - \ell - 1) + (2\ell + 8)} = \lim_{\Delta \rightarrow \infty} \frac{\Delta + \ell + 7}{2\Delta + 6} = \frac{1}{2}.$$

It remains to show the detailed construction of B during the regular game and the end game. We start with the regular game, where we proceed inductively. Consider round i and assume that previously only nodes not contained in G' have been matched by A . (The assumption holds in the first round.) All nodes in G' have degree at least three. Adversary B chooses the set of *allowed* degrees $D = \{3, \dots, \Delta\}$, cf. Section 5.3.1, and presents the highest priority data item $\langle P_1, \dots, P_k \rangle$ for which in each path $P_i = ((v_j^i, d_j^i, b_j^i))_{j=1}^{n_i}$ all degrees d_j^i are contained in D . Bits b_j^i are ignored, since the construction asserts that no edges incident with nodes in the data item have been removed earlier, as we show below.

For each path P_i adversary B adds the nodes v_j^i and edges $\{v_j^i, v_{j+1}^i\}$ contained in P_i as well as an edge connecting the last node $v_{n_i}^i$ with the first node v_1^i , see Figure 41. Furthermore, an edge connects each “odd” v -node $v_1^i, v_3^i, \dots, v_{n_i-1}^i$ with a u -node and each “even” v -node $v_2^i, v_4^i, \dots, v_{n_i}^i$ with a w -node, see double drawn edges in the figure. These edges are contained in a maximum matching, and each such edge connects with a u -node resp. with a w -node which does *not yet* have an incident optimal edge. Observe that, after edges are picked, the inductive hypothesis holds in the next round.

Adversary B has to deliver on its promise that node v_j^i has degree d_j^i . We only discuss an “odd” v -node, the argument is analogous for “even” v -nodes. Since v_j^i has two

incident path edges and one incident optimal edge, adversary B inserts $d_j^i - 3$ additional edges incident with v_j^i . In particular, these edges connect v_j^i with u -nodes. The degree constraint is maintained for u -nodes. Why? At most $d_j^i - 3 \leq \Delta - 3$ edges are added for each v -node, at most $\Delta - r \leq \Delta - 2$ many v -nodes need additional edges, there are Δ many u -nodes, each u -node can accept $\Delta - 4$ additional edges, and for sufficiently large Δ we have

$$(\Delta - 3)(\Delta - 2) = \Delta^2 - 5\Delta + 6 \leq \Delta^2 - 4\Delta = \Delta(\Delta - 4).$$

In the end game, adversary B adds a complete bipartite subgraph of $2r$ new nodes. Each new “odd” node is connected with an optimal edge to a distinct u -node. The degree constraint holds for u -nodes. Why? Since r many u -nodes did not receive an optimal edge during the regular game and we assumed above that each u -node receives at most $\Delta - 4$ edges from v -nodes. Analogously, each new “even” node is connected with an optimal edge to a distinct w -node without violating the degree constraint.

Since the subgraph is complete with at least $r \geq 2$ nodes in each partition and each node has an incident optimal edge, all added nodes have degree at least three. Consequently, adversary B acts consistently when presenting only nodes of degree at least three during the regular game. \square

Note. This construction also allows to show the same bound if an algorithm may additionally choose the partition of each node.

Hypergraph Matching. We conclude the proof section and show that the greedy approach fails for hypergraph matching.

Theorem 70. *Let $A \in \mathcal{VA}^k$ be an algorithm for matching on k -uniform hypergraphs. There is a graph on which A computes a matching of size exactly $\frac{1}{k}$ times optimal. In particular, in this graph degrees are bounded by at most four.*

Proof. In the adaptive priority game, the instance constructed by an adversary B is illustrated in Figure 42. White vertical edges constitute a maximum matching $\{e_0, \dots, e_{k-1}\}$ of size k . By labeling nodes in adversarial fashion, depending on the first priority order submitted by A , adversary B forces A to pick the topmost horizontal white edge e . We call a node an e -node if it belongs to e and a *non- e -node* otherwise. Further edges—see below—all contain an e -node. Hence no edge besides e can be picked by A and the approximation ratio is exactly $\frac{1}{k}$.

Observe that these requirements are satisfied by the construction given in Figure 43: the “matrix” is composed of an “upper triangular matrix” U , which contains nodes $1, \dots, K$ in row-wise ascending order, and a “lower triangular matrix” L , whose j -th column contains all nodes in the j -th row of U . Now, the i -th new edge, with $0 \leq i \leq k-2$, contains node v_i , the e -node of $e_{(i+1) \bmod (k-1)}$, and all nodes in S_i .

We discuss the strategy of B . Therefore we first observe the following properties of the construction:

- i. The e -nodes of e_0, \dots, e_{k-2} have degree four.
- ii. All other nodes, including nodes in S_0, \dots, S_{k-2} , have degree two.
- iii. Any two edges have at most one node in common.
- iv. Edge e shares exactly one node with any other edge.

In the first round, from the order submitted by A adversary B presents the highest priority data item $\langle u, W_1, \dots, W_{D_u}, W_i \rangle$ with $D_u \in \{2, 4\}$ (which are the only degrees present in the graph, by **i.** and **ii.**), $W_j \cap W_l = \emptyset$ for $j \neq l$ (node u is the only node common to all incident edges, by **iii.**) and $|W_j| = k-1$ for all j (the graph is k -uniform). Algorithm A picks edge $\{u\} \cup W_i$.

First assume that $D_u = 4$ holds. Then B relabels nodes such that u is the e -node of e_0 . This is consistent, since this is the first data item revealed to A . Furthermore, adversary B relabels nodes such that we have $e = \{u\} \cup W_i$, i.e. algorithm A picks edge e . The matching is maximal by **iv.** In case $D_u = 2$ holds, adversary B relabels nodes such that u is the e -node of e_{k-1} , which has degree two by construction, and again lets the picked edge be $e = \{u\} \cup W_i$. \square

Note. It would be desirable to know if a $\frac{1}{k}$ -bound can also be shown for k -partite uniform hypergraphs.

6 Conclusions

We studied greedy algorithms which compute maximal matchings by repeatedly adding edges to the solution, in particular deterministic algorithms which pick edges incident with nodes of current minimum degree. The KARSIPSER algorithm and 1-2-GREEDY, though conceptually very simple, achieve optimal approximation ratio on bipartite resp. general graphs of bounded degree. Optimality holds w.r.t. large classes of resource-unconstrained adaptive priority algorithms.

Proof Method. In the proof of the approximation guarantees for both algorithms we use a charging scheme based on connected components of the graph $(V, M \cup M^*)$, where M is the solution of the algorithm and M^* is a maximum matching.

A downside of the analysis is a quite detailed argument. Is there another approach which allows to prove the same bounds in a more concise analysis? If not, are there algorithms on par with KARSIPSER and MINGREEDY which allow a simpler analysis? This might be achieved for algorithms with more fine-grained control over which edge to pick next.

Stronger Data Items. For an adaptive priority algorithm, control over the next edge is restricted by the type of data items: a data item for node u might expose a neighbor of u , the degree of u , or even the entire set of neighbors of u . We have shown that even algorithms using the latter data items, namely $\mathcal{W}\mathcal{A}$ -algorithms, cannot beat the approximation performance of 1-2-GREEDY on general graphs, even though they are able to “traverse” the graph.

Our construction inherently depends on picking edges in triangles. If a $\mathcal{W}\mathcal{A}$ -like data item exposes the neighbors of *both* nodes of an edge, then an algorithm is able to prevent picking edges in triangles by requesting nodes with disjoint sets of neighbors. Is there an algorithm avoiding triangles which attains at least equal approximation performance? Can it even beat the performance of 1-2-GREEDY?

For bipartite graphs, a similar question remains to be answered. Here, we show that among degree sensitive \mathcal{DS}_{01} -algorithms the KARSIPSER algorithm is optimal with approximation ratio $\frac{\Delta}{2\Delta-2}$, if degrees are bounded by Δ . However, our inapproximability bound for $\mathcal{W}\mathcal{A}$ -algorithms merely converges to $\frac{3}{4}$ in our construction for $\Delta \rightarrow \infty$.

Is there a $\mathcal{W}\mathcal{A}$ -algorithm with stronger approximation performance than algorithms in \mathcal{DS}_{01} ? Are $\mathcal{W}\mathcal{A}$ -algorithms limited by factor $\frac{1}{2}$ as well?

Approaches to Randomization. If all ties in 1-2-GREEDY are broken uniformly at random, and degrees are bounded by at most Δ , is an expected approximation ratio larger than $\frac{\Delta-1}{2\Delta-3}$ achieved? Does this algorithm beat factor $\frac{1}{2}$ for $\Delta \rightarrow \infty$?

The randomized MRG algorithm achieves approximation ratio at least $\frac{1}{2} + \frac{1}{400.000}$, see [ADFS95]. What is the expected performance of an algorithm which picks an edge like 1-2-GREEDY if there is a node of degree at most two and like MRG otherwise?

References

- [AB04] Spyros Angelopoulos and Allan Borodin. The Power of Priority Algorithms for Facility Location and Set Cover. *Algorithmica*, 40(4):271–291, 2004.
- [ABMP91] Helmut Alt, Norbert Blum, Kurt Mehlhorn, and Markus Paul. Computing a Maximum Cardinality Matching in a Bipartite Graph in Time $O(n^{1.5}\sqrt{m/\log n})$. *Inf. Process. Lett.*, 37(4):237–240, 1991.
- [ADFS95] Jonathan Aronson, Martin Dyer, Alan Frieze, and Stephen Suen. Randomized Greedy Matching II. *Random Structures & Algorithms*, 6(1):55–73, 1995.
- [AFP98] Jonathan Aronson, Alan Frieze, and Boris G. Pittel. Maximum Matchings in Sparse Random Graphs: Karp–Sipser Revisited. *Random Structures & Algorithms*, 12(2):111–177, 1998.
- [AV79] Dana Angluin and Leslie G. Valiant. Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings. *J. Comput. Syst. Sci.*, 18(2):155–193, 1979.
- [BB12] Patrick Bennett and Tom Bohman. A Natural Barrier in Random Greedy Hypergraph Matching. *CoRR*, abs/1210.3581, 2012.
- [BBLM10] Allan Borodin, Joan Boyar, Kim S. Larsen, and Nazanin Mirmohammadi. Priority Algorithms for Graph Optimization Problems. *Theor. Comput. Sci.*, 411(1):239–258, 2010.
- [Ber57] Claude Berge. Two Theorems in Graph Theory. *Proc. Natl. Acad. Sci. USA*, 43(9):842–844, 1957.
- [Bes14] Bert Besser. Approximation Bounds for Minimum Degree Matching. *CoRR*, 2014. arXiv:1408.0596 [cs.DS].
- [BIYZ12] Allan Borodin, Ioana Ivan, Yuli Ye, and Bryce Zimny. On Sum Coloring and Sum Multi-Coloring for Restricted Families of Graphs. *Theor. Comput. Sci.*, 418:1–13, 2012.
- [Blu15] Norbert Blum. Maximum Matching in General Graphs Without Explicit Consideration of Blossoms Revisited. *CoRR*, 2015. arXiv:1509.04927 [cs.DS].
- [BNR03] Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (Incremental) Priority Algorithms. *Algorithmica*, 37(4):295–326, 2003.

- [BP15] Bert Besser and Matthias Poloczek. Greedy Matching: Guarantees and Limitations. *Algorithmica*, 2015. To appear, <http://dx.doi.org/10.1007/s00453-015-0062-2>.
- [BSX08] Bonnie Berger, Rohit Singht, and Jinbo Xu. Graph Algorithms for Biological Systems Analysis. In *SODA*, pages 142–151, 2008.
- [BW15] Bert Besser and Bastian Werth. On the Approximation Performance of Degree Heuristics for Matching. *CoRR*, 2015. arXiv:1504.05830 [cs.DS].
- [CCWZ14] T.-H. Hubert Chan, Fei Chen, Xiaowei Wu, and Zhichao Zhao. Ranking on Arbitrary Graphs: Rematch via Continuous LP with Monotone and Boundary Condition Constraints. In *SODA*, pages 1112–1122, 2014.
- [CL12] Yuk Hei Chan and Lap Chi Lau. On Linear and Semidefinite Programming Relaxations for Hypergraph Matching. *Math. Program.*, 135(1-2):123–148, 2012.
- [CWC⁺96] Yong-Qing Cheng, Victor Wu, Robert T. Collins, Allen R. Hanson, and Edward M. Riseman. Maximum-Weight Bipartite Matching Technique and Its Application in Image Feature Matching. In *In Proc. SPIE Visual Comm. and Image Processing*, 1996.
- [Cyg13] Marek Cygan. Improved Approximation for 3-Dimensional Matching via Bounded Pathwidth Local Search. In *FOCS*, pages 509–518, 2013.
- [DF91] Martin E. Dyer and Alan M. Frieze. Randomized Greedy Matching. *Random Structures & Algorithms*, 2(1):29–46, 1991.
- [DFP93] Martin Dyer, Alan Frieze, and Boris Pittel. The Average Performance of the Greedy Matching Algorithm. *The Annals of Applied Probability*, 3(2):526–552, 1993.
- [DI09] Sashka Davis and Russell Impagliazzo. Models of Greedy Algorithms for Graph Problems. *Algorithmica*, 54(3):269–317, 2009.
- [Edm65] Jack Edmonds. Paths, Trees, and Flowers. *Canad. J. Math.*, 17:449–467, 1965.
- [FRS95] Alan M. Frieze, Andrew J. Radcliffe, and Stephen Suen. Analysis of a Simple Greedy Matching Algorithm on Random Cubic Graphs. *Combinatorics, Probability & Computing*, 4:47–66, 1995.

- [Gab76] Harold N. Gabow. An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs. *J. ACM*, 23(2):221–234, 1976.
- [Gab14] Harold N. Gabow. Set-merging for the Matching Algorithm of Micali and Vazirani. *CoRR*, 2014. arXiv:1501.00212 [cs.DS].
- [GH90] Olivier Goldschmidt and Dorit S. Hochbaum. A Fast Perfect-Matching Algorithm in Random Graphs. *SIAM J. Discrete Math.*, 3(1):48–57, 1990.
- [GK04] Andrew V. Goldberg and Alexander V. Karzanov. Maximum Skew-Symmetric Flows and Matchings. *Math. Program.*, 100(3):537–568, 2004.
- [GT85] Harold N. Gabow and Robert Endre Tarjan. A Linear-Time Algorithm for a Special Case of Disjoint Set Union. *J. Comput. Syst. Sci.*, 30(2):209–221, 1985.
- [GT91] Harold N. Gabow and Robert Endre Tarjan. Faster Scaling Algorithms for General Graph-Matching Problems. *J. ACM*, 38(4):815–853, 1991.
- [GT12] Gagan Goel and Pushkar Tripathi. Matching with Our Eyes Closed. In *FOCS*, pages 718–727, 2012.
- [Har09] Nicholas J. A. Harvey. Algebraic Algorithms for Matching and Matroid Problems. *SIAM J. Comput.*, 39(2):679–702, 2009.
- [HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.
- [Hou09] Stefan Hougardy. Linear Time Approximation Algorithms for Degree Constrained Subgraph Problems. In William Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 185–200. Springer, 2009.
- [HS89] C. A. J. Hurkens and A. Schrijver. On the Size of Systems of Sets Every T of Which Have an SDR, with an Application to the Worst-case Ratio of Heuristics for Packing Problems. *SIAM J. Discret. Math.*, 2(1):68–72, 1989.
- [HS07] Madhusudan Hosaagrahara and Harish Sethu. Degree-Sequenced Matching Algorithms for Input-Queued Switches. *Telecommunication Systems*, 34(1-2):37–49, 2007.

- [HSS06] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the Complexity of Approximating k-Set Packing. *Comput. Complex.*, 15(1):20–39, 2006.
- [KH78] Bernhard Korte and Dirk Hausmann. An Analysis of the Greedy Heuristic for Independence Systems. In *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 65–74. , 1978.
- [KK98] George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal On Scientific Computing*, 20(1):359–392, 1998.
- [KS81] Richard M. Karp and Michael Sipser. Maximum Matchings in Sparse Random Graphs. In *FOCS*, pages 364–375, 1981.
- [KVV90] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An Optimal Algorithm for On-line Bipartite Matching. In *STOC*, pages 352–358, 1990.
- [LGXB11] Bo Lu, Robert Greevy, Xinyi Xu, and Cole Beck. Optimal Nonbipartite Matching and Its Statistical Applications. *The American Statistician*, 65(1):21–30, 2011.
- [LMS10] Johannes Langguth, Fredrik Manne, and Peter Sanders. Heuristic Initialization for Bipartite Matching Problems. *ACM Journal of Experimental Algorithmics*, 15, 2010.
- [LP86] László Lovász and Michael David Plummer. *Matching Theory*. North-Holland, 1986.
- [Mag98] Jacob Magun. Greedy Matching Algorithms: An Experimental Study. *ACM Journal of Experimental Algorithmics*, 3:6, 1998.
- [MMH95] Rolf H. Möhring and Matthias Müller-Hannemann. Cardinality Matching: Heuristic Search for Augmenting Paths. Technical report, Technische Universität Berlin, 1995.
- [MP97] Zevi Miller and Dan Pritikin. On Randomized Greedy Matchings. *Random Struct. Algorithms*, 10(3):353–383, 1997.
- [MS04] Marcin Mucha and Piotr Sankowski. Maximum Matchings via Gaussian Elimination. In *FOCS*, pages 248–255, 2004.

- [MSVV07] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. AdWords and Generalized Online Matching. *J. ACM*, 54(5), 2007.
- [MV80] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matchings in General Graphs. In *FOCS*, pages 17–27, 1980.
- [PF90] Alex Pothén and Chin-Ju Fan. Computing The Block Triangular Form of a Sparse Matrix. *ACM Trans. Math. Softw.*, 16(4):303–324, 1990.
- [Pol11] Matthias Poloczek. Bounds on Greedy Algorithms for MAX SAT. In *ESA*, pages 37–48, 2011.
- [Pol12] Matthias Poloczek. *Greedy Algorithms for MAX SAT and Maximum Matching: Their Power and Limitations*. PhD thesis, Institut für Informatik, Goethe-Universität Frankfurt am Main, 2012.
- [PS12] Matthias Poloczek and Mario Szegedy. Randomized Greedy Algorithms for the Maximum Matching Problem with New Analysis. In *FOCS*, pages 708–717, 2012.
- [RSL77] Daniel J. Rosenkrantz, Richard Edwin Stearns, and Philip M. Lewis II. An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM J. Comput.*, 6(3):563–581, 1977.
- [RSÜ05] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Pairwise Kidney Exchange. *Journal of Economic Theory*, 125(2):151–188, 2005.
- [Sha97] Andrew Shapira. An Exact Performance Bound for an $O(m+n)$ Time Greedy Matching Procedure. *The Electronic Journal of Combinatorics*, 4(1):R25, 1997.
- [Tin84] Gottfried Tinhofer. A Probabilistic Analysis of some Greedy Cardinality Matching Algorithms. *Annals of Operations Research*, 1(3):239–254, 1984.
- [Tri12] Pushkar Tripathi. *Allocation Problems with Partial Information*. PhD thesis, Georgia Institute of Technology, 2012.
- [Vaz13] Vijay V. Vazirani. A Simplification of the MV Matching Algorithm and its Proof. *CoRR*, 2013. arXiv:1210.4594 [cs.DS].
- [Vit85] Jeffrey Scott Vitter. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

Appendix A. Linear Time Implementations

Literature on randomized greedy matching frequently states that a given algorithm can be implemented in linear time $O(|V| + |E|)$ for input graph $G = (V, E)$, but does not present such an implementation. In this section we devise data structures for several algorithms.

In any case, we do not claim to be the first to present such an implementation, and we provide references to the literature to the best of our knowledge. For MRG and SHUFFLE, see [PS12].

Preparations. We assume that the input graph $G = (V, E)$ is defined on node set $V = \{0, 1, \dots, n-1\}$ and that G is given in the default *adjacency list* representation, i.e. for any node x a list L_x contains all neighbors of x .

In a linear time preprocessing step, we compute *linked adjacency arrays*. Their purpose is to allow the removal of any given edge from the graph in constant time. The linked adjacency array A_x for node x stores all neighbors of x and has the same initial length as L_x . During the course of the algorithm the number of cells in A_x does not shrink but we assert that in each step the remaining neighbors of x are stored in a consecutive prefix of A_x , and the rest of A_x is empty. We track the size of the non-empty part of A_x using a counter.

How can an edge be removed in constant time? Let a node x and an index of a cell in A_x be given, say containing neighbor y . The array cell in A_x containing node y also stores a *link to x in A_y* , i.e. the index of the array cell in A_y containing node x , see Figure 44. In order to remove the edge $\{x, y\}$, we move the entry of the last non-empty cell in A_x to the position of y , and proceed analogously for A_y and x . We also update the links of the two moved entries accordingly; this is done in constant time using the links stored inside the moved entries.

To initialize linked adjacency arrays in linear time $O(|V| + |E|)$, we first initialize empty arrays A_0, \dots, A_{n-1} of lengths $|L_0|, \dots, |L_{n-1}|$, respectively. Then we scan each of L_0, \dots, L_{n-1} in this order from beginning to end. When edge $\{x, y\}$ occurs for the first time then we save a linked pair of entries in the currently first empty cells in A_x and A_y . We ignore the second occurrence of edge $\{x, y\}$ like this: when we are scanning list L_x and encounter $\{x, y\}$ with $x < y$, then we process $\{x, y\}$ as described above; when we scan list L_y and encounter $\{x, y\}$ when $y > x$ holds, then $\{x, y\}$ was already processed earlier and we ignore the edge.

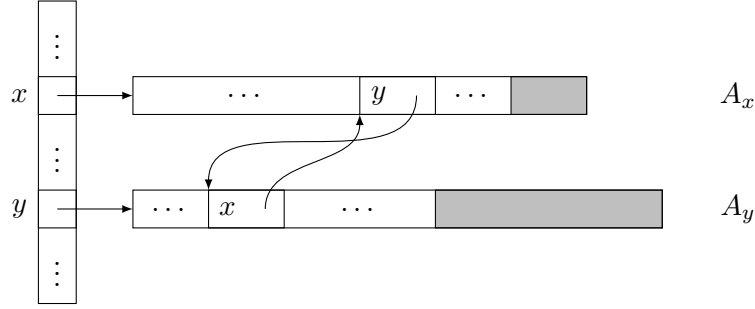


Figure 44: Linked adjacency arrays with an example edge $\{x, y\}$: the gray sub-arrays are already empty

A.1 GREEDY

Additional to linked adjacency arrays, we use an *edge array* of length $|E|$. In a linear time preprocessing phase, the edge array is filled with all edges $\{i, j\} \in E$, in arbitrary order (when scanning adjacency lists we ignore the second occurrence of each edge like we do in the construction of linked adjacency arrays). For any given edge $\{i, j\}$, both representations of $\{i, j\}$ in the linked adjacency arrays of nodes i and j as well as the representation of $\{i, j\}$ in the edge array also contain a link to the other representations of $\{i, j\}$.

During the course of the algorithm, a counter m initialized with $m = |E|$ tracks the number of edges left in the graph, where the edge array stores the m remaining edges in a consecutive sub-array beginning at the first cell.

In each step, a random edge $\{i, j\}$ is picked from the first m non-empty cells in the edge array.

When removing edges incident with nodes i and j by scanning their respective linked adjacency arrays, the edge array representation of each edge can be removed by following a link. Each time that an edge is removed from the edge array, the empty cell is filled by moving the currently last edge in the edge array. Links towards the moved edge are updated by following links from the moved edge.

Runtime. As discussed above, building linked adjacency arrays and the edge array takes linear time $O(|V|+|E|)$. During the GREEDY algorithm, at most $|M| = O(|V|)$ edges are picked and removed from the edge array (each in constant time), the adjacency array of each node is scanned at most once, and each of the $|E|$ edges is removed at most once from the linked adjacency arrays and from the edge array (each in constant time).

A.2 MINGREEDY and DOUBLE-MINGREEDY

Implementations of MINGREEDY based on priority queues were sketched in [LMS10] and [Sha97]. However, using a heap to maintain lists of nodes—one list for each degree in the current graph—uses time $\log_2 |V|$ in each iteration. Therefore the runtime is at least $\Omega(|V| \log_2 |V| + |E|)$.

A brief description of a linear time implementation of MINGREEDY can be found in [Mag98]. We present in detail an $O(|V| + |E|)$ time implementation of MINGREEDY. Furthermore, we also discuss how to use our data structure to implement DOUBLE-MINGREEDY in $O(|V| + |E|)$ time.

Runtime. Our data structure can be initialized in linear time $O(|V| + |E|)$ by scanning through the adjacency lists of G a constant number of times. At any time during MINGREEDY, the data structure supports each of the following operations in constant time: selection of a random node of minimum (non-zero) degree, selection of a random neighbor of a given node, and the deletion of a given edge. Hence MINGREEDY can be implemented in linear time since a minimum degree node and a neighbor are selected at most $\frac{|V|}{2}$ times and each of the $|E|$ edges is removed exactly once.

Implementation. How can a minimum degree node u be selected in constant time? Consider a step of MINGREEDY and let $d_0 < d_1 < \dots < d_k$ be the different degrees currently present in the graph, where $d_0 = 0$ is the degree of already isolated nodes. We use an array S which is partitioned into sub-arrays S_i with $0 \leq i \leq k$ such that S_i precedes all S_j with $i < j$. Each S_i contains all nodes with current degree d_i in contiguous cells of S . A doubly linked list D stores, from head to tail, the borders $(l_0, r_0), (l_1, r_1), \dots, (l_k, r_k)$ of S_0, S_1, \dots, S_k , respectively, see Figure 45. A minimum degree node u is selected by reading the second entry in D , which stores nodes of minimum degree d_1 , and choosing a random cell in S_1 . A random neighbor of u can be selected in constant time from the non-empty part of the adjacency array A_u of u .

How to update S and D when removing an edge $\{x, y\}$? Since the degrees d_x and d_y of x resp. y are decreased by exactly one, these nodes are moved from sub-array S_{d_x} to sub-array S_{d_x-1} respectively from S_{d_y} to S_{d_y-1} . We proceed analogously for x and y : to move x to its new sub-array, we utilize two helper arrays P_D and P_S . The cell $P_D[x]$ stores a pointer to the D -entry of the sub-array of S containing x , the cell $P_S[x]$ holds the index of the cell in S containing x . The entry of node x in the sub-array S_{d_x} is replaced by the “leftmost” node in S_{d_x} , call it z , i.e. node z moves from the smallest index in S_{d_x} to the now empty cell of x , and x is appended to the “right” of S_{d_x-1} . In particular,

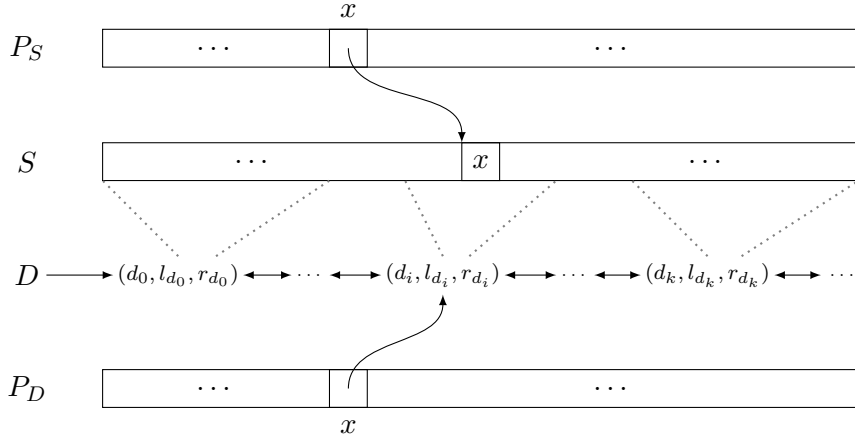


Figure 45: The “degree-tracking” data structure with an example node x of degree d_i

array S is still partitioned into sub-arrays S_0, \dots, S_k . If S_{d_x} is now empty, we remove it from D in constant time using the pointer $P_D[x]$. If S_{d_x-1} does not yet exist, i.e. node x is now the only node of degree $d_x - 1$, then we create S_{d_x-1} at the now cleared position in S and insert S_{d_x-1} before S_{d_x} (or before the successor of S_{d_x} , if S_{d_x} was removed), also in constant time. The pointers in P_D and the addresses stored in P_S are updated accordingly, for x as well as for z .

To construct S, D, P_S , and P_D during a linear time preprocessing phase, we first scan the linked adjacency arrays to compute, for each possible degree d in G , a list of nodes having degree d . There are at most $|V|$ such lists. Scanning these lists from smallest to largest degree, we can fill S and D from beginning to end and set pointers in P_S and P_D accordingly.

DOUBLE-MINGREEDY. After picking a random minimum degree node u , to find a random minimum degree neighbor we scan the neighbors of u in the linked adjacency list A_u once. Therefore we employ reservoir sampling, see Algorithm R in [Vit85]. In particular, using a reservoir of size one, we sample only from the minimum degree neighbors of u , where we reinitialize the reservoir as well as the sampling process each time that we encounter a neighbor with degree smaller than the currently known minimum degree of a neighbor of u . The sampling takes time $O(|A_u|)$. This time is asymptotically not larger than the time needed to remove the edges incident with node u from the graph, therefore the total time remains linear.

A.3 The KARPSSIPER Algorithm and 1-2-GREEDY

We combine our implementations for GREEDY and MINGREEDY, i.e. we use linked adjacency arrays, the edge array defined for GREEDY, as well as the arrays and lists S , D , P_S , and P_D defined for MINGREEDY.

In each round we can decide in constant time if a node of current degree one exists by inspecting the second element in the doubly linked list D . If so, then we can pick the edge incident with a random degree-1 node as described for MINGREEDY. If not, then we pick an edge as described for GREEDY. The update of all data structures is accomplished by removing edges incident with the newly matched nodes one by one. Since for each data structure the removal of a single edge is an encapsulated transformation, which brings the respective data from one consistent state into the next, both implementations can be combined without effort. Moreover, linear runtime carries over to the combined data structure.

To implement 1-2-GREEDY, we proceed analogously but in each step we test for the existence of a node of degree at most two.

DEGREE HEURISTICS FOR MATCHING

VON
BERT BESSER

Sogenannte Matching-Probleme finden sich in einer Vielzahl von Anwendungen wie z. B. der Bilderkennung [CWC⁺96], der Partitionierung von Rechenclustern [KK98], der Suche nach kompatiblen Organspendern [RSÜ05, Tri12], dem Routing von Netzwerkverkehr [HS07] oder dem Vergleich von Proteinstrukturen [BSX08].

Im Kern solcher Anwendungen wird beispielsweise nach einer Zuweisung von Prozessoren zu Rechenclustern, Organspendern zu Patienten oder eingehendem Netzwerkverkehr zu Ausgabeports gesucht. Diese Zuweisungen können als Mengen von knotendisjunkten Kanten modelliert werden, wobei die Anwendung die Eigenschaften des zugrundeliegenden Graphen bestimmt (z. B. Bipartitheit oder Gewichtung).

In dieser Arbeit untersuchen wir das grundlegendste aller Matching-Probleme, das sogenannte *Maximum Cardinality Matching Problem*: In einem ungerichteten ungewichteten Graphen soll eine größtmögliche Menge von knotendisjunkten Kanten identifiziert werden.

Greedy-Algorithmen. Zur Approximation dieses Problems existiert eine Vielzahl sogenannter Greedy-Algorithmen. Solche Algorithmen sind konzeptionell ungleich einfacher als bekannte Polynomialzeitverfahren und in Szenarien mit beschränktem Zugriff auf den Eingabegraphen [RSÜ05, Tri12] alternativlos.

Ein Greedy-Algorithmus startet mit einem leeren Matching, welchem wiederholt eine Kante hinzugefügt wird, die zwei bis dahin ungematchte Knoten miteinander verbindet:

$M \leftarrow \emptyset$	▷ initialisiere ein leeres Matching
while $E \neq \emptyset$ do	▷ wiederhole solange noch Kanten existieren
wähle eine Kante $\{u, v\} \in E$	▷ <i>Kantenwahl-Heuristik</i> hier einsetzen
$M \leftarrow M \cup \{\{u, v\}\}$	▷ nimm Kante $\{u, v\}$ ins Matching auf
$E \leftarrow E \setminus \{\{x, y\} : x \in \{u, v\}, y \in V\}$	▷ entferne mit u und v inzidente Kanten
end while	
return M	

Wird dem Matching eine Kante hinzugefügt, so wird sie später nicht wieder entfernt. Der Algorithmus terminiert, sobald keine Kante mehr aufgenommen werden kann, ohne dass ein Knoten zweimal gematcht wird. Im Allgemeinen ist das ausgegebene Matching nicht größtmöglich.

Bekannte Resultate. Einige Greedy-Ansätze für das Matching-Problem wurden für den algorithmischen Nachweis großer Matchings in Zufallsgraphen entwickelt: Anders als in einem nicht-konstruktiven Beweis wird gezeigt, dass ein gegebener Algorithmus fast sicher ein großes Matching errechnet. Für große Zufallsgraphen liefern Greedy-Algorithmen in der Regel sehr gute bis optimale Approximationen [KS81, Tin84, GH90, DFP93, FRS95, AFP98].

Hingegen bestehen für allgemeine Graphen deutliche Leistungsunterschiede: Da alle Greedy-Algorithmen mindestens Approximationsfaktor $\frac{1}{2}$ erreichen [KH78], stellt sich die Frage, welche Algorithmen die $\frac{1}{2}$ -Schranke zu durchbrechen vermögen.

Für einige Greedy-Algorithmen können Familien von Graphen konstruiert werden, welche den Approximationsfaktor beliebig nahe nach $\frac{1}{2}$ konvergieren lassen [DF91, Pol12, BP15]. Für Algorithmen, die Faktor $\frac{1}{2}$ übertreffen, sind scharfe Ergebnisse nicht bekannt und große Lücken müssen geschlossen werden [ADFS95, CCWZ14]: Beispielsweise weiß man über den Approximationsfaktor des MRG-Algorithmus nur, dass er im Intervall $[\frac{1}{2}+c, \frac{2}{3}]$ für eine kleine Konstante c liegt.

Folglich wendete sich die Forschung eingeschränkten Graphklassen zu, wie beispielsweise Bäumen, planaren Graphen oder Graphen mit großem Girth. Insbesondere für gradbeschränkte Graphen wurden stärkere Resultate erzielt [DF91, ADFS95, MP97]; Jedoch waren Schranken für “Grad-Heuristiken” wie KARPSSIPSE oder 1-2-GREEDY lange unbekannt.

Neben dem Bestreben, einen gegebenen Algorithmus vollständig zu verstehen, wird erforscht, welche Eigenschaften eines Algorithmus seine Stärke ausmachen. Dafür wurden für ganze Klassen von Algorithmen Schranken der Approximationskraft ermittelt [GT12, Pol12, BP15]. Insbesondere ist mit dem Greedy-Ansatz die Erzeugung optimaler Matchings im Allgemeinen nicht möglich [Pol12].

Fokus der Arbeit. In einer systematisch Studie untersuchen wir deterministische Greedy-Algorithmen für das Maximum Cardinality Matching Problem. Dabei legen wir unser Hauptaugenmerk auf die Stärken und Schwächen von Grad-Heuristiken, unter denen wir uns insbesondere den folgenden widmen.

- Der sogenannte KARPSSIPER-Algorithmus ist benannt nach seinen Autoren [KS81]. KARPSSIPER wählt, falls möglich, eine Kante mit einem Knoten von Grad Eins: eine solche Kante gehört zu einem optimalen Matching, daher ist ein solcher Schritt “optimal”. Andernfalls wird irgendeine Kante gewählt.
- Der 1-2-GREEDY-Algorithmus ist eine Spezialisierung von KARPSSIPER. Falls ein Knoten mit Grad höchstens Zwei existiert, wählt 1-2-GREEDY zuerst einen Knoten minimalen Grades und anschließend einen beliebigen Nachbarn. Andernfalls wird irgendeine Kante gewählt.

1-2-GREEDY kann auch als vereinfachte Variante des bekannten MINGREEDY-Algorithmus [Tin84] verstanden werden.

Wir analysieren die Approximationsleistung beider Ansätze und vergleichen sie mit bekannten Algorithmen aus der Literatur wie auch mit konzeptionell einfacheren und komplexeren hypothetischen Algorithmen: 1-2-GREEDY erreicht Approximationsfaktor

$$\frac{\Delta - 1}{2\Delta - 3}$$

wenn Grade durch höchstens Δ beschränkt sind; Wenn der Graph bipartit ist, dann erreicht der eingeschränkte KARPSSIPER-Algorithmus sogar den Faktor

$$\frac{\Delta}{2\Delta - 2}$$

Insbesondere übertreffen diese *Garantien* die besten bekannten Schranken für den *erwarteten* Approximationsfaktor randomisierter Greedy-Algorithmen.

Informationen über die Eingabe. Im Vergleich von KARPSSIPER und 1-2-GREEDY mit hypothetischen Algorithmen konzentrieren wir uns auf die Kantenwahl-Heuristik. Die Kantenwahl-Heuristik kann umso ausgereifter sein, je mehr Informationen über den Eingabegraphen $G = (V, E)$ vorliegen. Das Wissen über die Eingabe wird in jedem Schritt eines Algorithmus erweitert: Wenn beispielsweise festgestellt wird, dass die Kante $\{u, v\}$ nicht existiert, dann kann geschlossen werden, dass die Knoten u und v Grad höchstens $|V| - 2$ haben; Der MINGREEDY-Algorithmus berechnet anfangs die Grade aller Knoten und aktualisiert diese in jedem Schritt nachdem Kanten aus dem Graphen entfernt wurden; Wenn der SHUFFLE-Algorithmus einen Knoten auf Nachbarn testet, dann gewinnt SHUFFLE Kenntnis über nicht im Graphen enthaltene Kanten.

Typischerweise verarbeitet ein Algorithmus nicht alle gewonnenen Informationen. Beispielsweise ignoriert der GREEDY-Algorithmus jegliche Information und wählt in jedem Schritt *irgendeine* der verbleibenden Kanten.

Darüber hinaus ist in einigen Szenarien nur eingeschränkter Zugriff auf die Eingabe möglich und der Informationsgewinn entsprechend begrenzt. Beispielsweise kann ein Algorithmus für das sogenannte *Query Commit Problem* [RSÜ05, Tri12] keine Knotengrade lesen, da stets die Existenz nur genau *einer* Kante getestet wird.

Informationen nutzen. Die Menge und Art der über den Eingabegraphen gewonnenen Informationen limitiert die Stärke der Kantenwahl-Heuristik, welche wiederum die Approximationsgüte der Lösung bestimmt. Wir untersuchen systematisch, welche Informationen für eine starke Approximationsgüte unerlässlich sind; Weiterhin erforschen wir, welche Güte erreicht werden kann, wenn die pro Schritt zu gewinnenden Informationen beschränkt sind. Hierzu konzentrieren wir uns in erster Linie auf zwei Fragestellungen.

1. *Welche Approximationskraft ist erreichbar, wenn Informationen über Knotengrade auf beliebige Weise verarbeitet werden dürfen? Welche solche “Grad-Heuristik” liefert die stärksten Resultate?*
2. *Steigt die Approximationskraft unter Verwendung von Informationen über Nachbarn von gematchten Knoten? Anders als Grad-Heuristiken sind solche Algorithmen in der Lage den Eingabegraphen zu “traversieren”.*

Zur Klärung dieser Fragen verwenden wir das Framework der *Adaptive Priority Algorithms* von Borodin, Nielsen und Rackoff [BNR03]: In jeder Runde fragt ein adaptiver Priorityalgorithmus eine oder mehrere Kanten an, indem deren Eigenschaften – beispielsweise “ist inzident mit einem Knoten kleinsten Grades” – formuliert werden. Die erhaltenen Kanten werden zur Lösung hinzugefügt. Es gelten keine Beschränkungen für Rechenzeit und Speicherplatz; Daher sind adaptive Priorityalgorithmen ausschließlich durch ihre Eigenschaft limitiert Kanten “greedy-artig” zu wählen.

1. Wir zeigen, dass keine Grad-Heuristik auf Graphen mit Grad höchstens Δ den Approximationsfaktor $\frac{\Delta-1}{2\Delta-3}$ übertrifft. Also ist unsere Garantie für 1-2-GREEDY scharf und 1-2-GREEDY optimal unter allen Grad-Heuristiken.

Analog wird der Faktor $\frac{\Delta}{2\Delta-2}$ auf bipartiten Graphen nicht übertroffen und KARPSSIPER erweist sich als optimal. Selbst “doppelseitige” Strategien wie DOUBLE-MINGREEDY oder MDS übertreffen Faktor $\frac{\Delta}{2\Delta-2}$, wenn überhaupt, nur geringfügig.

Neben der Suche nach der besten Grad-Heuristik beschäftigen wir uns auch mit der folgenden grundlegenden Frage.

Gibt es eine Heuristik, die keine Grad-Informationen nutzt und hinsichtlich ihrer Approximationskraft MINGREEDY oder KARPSIPSER ebenbürtig ist?

Wir zeigen, dass selbst auf Pfaden (für die KARPSIPSER und 1-2-GREEDY optimale Matchings berechnen) der Faktor $\frac{1}{2}$ nicht übertroffen wird. Die Verarbeitung von Grad-Informationen ist also unverzichtbar, um nicht-triviale Schranken zu erreichen.

2. Algorithmen, welche auf die Verwendung von Grad-Informationen beschränkt sind, haben bei der Wahl einer Kante $\{u, v\}$ keine Kontrolle über Distanz oder Verbindung der Knoten u und v zu vorher gematchten Knoten. Es besteht daher die Gefahr, dass $\{u, v\}$ zwei Kanten eines optimalen Matchings “berührt”: viele solcher ungünstigen Wahlen bedeuten eine Konvergenz der Approximationsrate gegen $\frac{1}{2}$.

Dieser Gefahr wird in der folgenden Strategie begegnet. “Erzeuge” einen alternierenden Pfad, indem wiederholt eine Kante an einem seiner Enden ins Matching aufgenommen wird; falls der Pfad nicht vergrößert werden kann, erzeuge den nächsten Pfad. Für diese “Traversierung” des Graphen ist der Algorithmus auf Informationen über die Nachbarn gematchter Knoten angewiesen.

Kann ein Algorithmus, der Nachbars-Informationen in beliebiger Weise verarbeiten darf, die Approximationsleistung von Grad-Heuristiken übertreffen?

Unsere Nichtapproximierbarkeitsschranke von $\frac{\Delta-1}{2\Delta-3}$ für nicht-bipartite Graphen gilt auch in dieser größeren Klasse von Algorithmen. Insbesondere ist 1-2-GREEDY optimal, obwohl Nachbars-Informationen nicht ausgenutzt werden.

Unsere Resultate zeigen also, dass unter Algorithmen mit nicht-trivialer Approximationsgarantie KARPSIPSER und 1-2-GREEDY durch ihre Vereinigung von konzeptioneller Einfachheit und optimaler Approximationsleistung herausstechen.

Weitere Ergebnisse. Klassische Greedy-Strategien ergänzen das Matching in jedem Schritt um eine einzelne Kante. Wir untersuchen auch den Ansatz, statt einzelner Kanten ganze (alternierende) Pfade zu wählen. Auch hier untersuchen wir entsprechende Klassen

von adaptiven Priorityalgorithmen und zeigen, dass trotz dieser Verallgemeinerung der Approximationsfaktor von KARPSSIPSE und 1-2-GREEDY nicht grundlegend verbessert wird: für $\Delta \rightarrow \infty$ gilt weiterhin die $\frac{1}{2}$ -Schranke.

Der Erfolg von Greedy-Algorithmen für Matching überträgt sich nicht auf das allgemeinere Matching-Problem in k -uniformen Hypergraphen (auch als k -Set Packing Problem bekannt). Wir zeigen, dass für $\Delta = O(1)$ der triviale Faktor $\frac{1}{k}$ nicht übertroffen wird, selbst wenn Nachbars-Informationen ausgenutzt werden dürfen.