

Greedy Matching: Guarantees and Limitations

Bert Besser*

Matthias Poloczek†

Abstract

Since Tinhofer proposed the MINGREEDY algorithm for maximum cardinality matching in 1984, several experimental studies found the randomized algorithm to perform excellently for various classes of random graphs and benchmark instances. In contrast, only few analytical results are known. We show that MINGREEDY cannot improve on the trivial approximation ratio of $\frac{1}{2}$ whp., even for bipartite graphs. Our hard inputs seem to require a small number of high-degree nodes.

This motivates an investigation of greedy algorithms on graphs with maximum degree Δ : We show that MINGREEDY achieves a $\frac{\Delta-1}{2\Delta-3}$ -approximation for graphs with $\Delta=3$ and for Δ -regular graphs, and a guarantee of $\frac{\Delta-1/2}{2\Delta-2}$ for graphs with maximum degree Δ . Interestingly, our bounds even hold for the deterministic MINGREEDY that breaks all ties arbitrarily.

Moreover, we investigate the limitations of the greedy paradigm, using the model of *priority algorithms* introduced by Borodin, Nielsen, and Rackoff. We study deterministic priority algorithms and prove a $\frac{\Delta-1}{2\Delta-3}$ -inapproximability result for graphs with maximum degree Δ ; thus, these greedy algorithms do not achieve a $\frac{1}{2}+\epsilon$ -approximation and in particular the $\frac{2}{3}$ -approximation obtained by the deterministic MINGREEDY for $\Delta=3$ is optimal in this class. For k -uniform hypergraphs we show a tight $\frac{1}{k}$ -inapproximability bound.

We also study fully randomized priority algorithms and give a $\frac{5}{6}$ -inapproximability bound. Thus, they cannot compete with matching algorithms of other paradigms.

1 Introduction

Due to their simplicity and efficiency, greedy algorithms have been studied intensely for the maximum cardinality matching problem, a problem that arises in many applications including image feature matching [11], pairwise kidney exchange [40, 42], protein structure comparison [5], and low delay network traffic routing [26].

In 1984 Tinhofer [41] proposed the following three randomized greedy algorithms. All have in common that iteratively an edge is picked and added to the matching; to obtain a feasible matching, afterwards both endpoints are removed from the graph with all their incident edges. Thus, the crucial aspect is how the edge is selected. The first algorithm, simply referred to as GREEDY, picks the edge uniformly at random among all edges. The modified randomized greedy algorithm (MRG), however, selects a node, the so-called first endpoint, uniformly at random and matches it to a randomly picked neighbor, the second endpoint. Tinhofer’s third algorithm, MINGREEDY, is identical,

*Institut für Informatik, Goethe-Universität Frankfurt am Main. Email: besser@thi.cs.uni-frankfurt.de. Partially supported by DFG SCHN 503/6-1.

†School of Operations Research and Information Engineering, Cornell University. Email: poloczek@cornell.edu. Supported by the Alexander von Humboldt Foundation within the Feodor Lynen program, and in part by NSF grant CCF-1115256.

except that the first endpoint is selected uniformly at random among all nodes that have minimum degree. These greedy algorithms can be implemented in linear time using only simple data structures; for MINGREEDY we describe such a data structure in Sect. A. A linear time implementation of MRG is proposed in [39].

Note that MINGREEDY can be interpreted as a refinement of MRG, since MINGREEDY prioritizes nodes that have a minimum number of matchable neighbors remaining. And indeed, Tinhofer provides experimental and analytical results for Erdős-Rényi random graphs with varying density: here MINGREEDY achieves an expected matching size larger than MRG or GREEDY. Frieze, Radcliffe, and Suen [18] found a superior performance of MINGREEDY on random cubic graphs: for instance, MINGREEDY left only about 10 out of 10^6 nodes unmatched, and hence performed better than MRG by orders of magnitude. An excellent performance was also observed in experiments for random graphs with small constant average degree [34] and on graphs arising from a real world application [26].

In contrast little is known about rigorous performance guarantees of MINGREEDY: The most important analytical result is due to Frieze, Radcliffe, and Suen [18] and states that MINGREEDY leaves only $o(n)$ nodes unmatched on random cubic graphs in expectation. No worst case analysis is known. On the other hand, MRG beats approximation ratio $\frac{1}{2}$ by a small constant [2, 39] on general graphs, whereas GREEDY cannot [15]. Recently, Chan, Chen, Wu, and Zhao [9] showed that the RANKING algorithm of [32] achieves at least a 0.523-approximation on general graphs; we refer to [35, 30] for results on bipartite graphs and to [39] for Erdős-Rényi random graphs.

Our first result is a worst case analysis of MINGREEDY (see Sect. 2): We propose a family of bipartite graphs and show that MINGREEDY does not achieve approximation ratio $\frac{1}{2} + \varepsilon$ whp. for any $\varepsilon > 0$. Thus, somewhat surprisingly, it performs even worse than the seemingly less optimized MRG algorithm on their respective hardest inputs. Moreover, we show that two popular randomized variants of MINGREEDY also fail on our graphs.

The closer the approximation ratio of MINGREEDY is pushed towards $\frac{1}{2}$, the larger the number of nodes n must be in order to obtain the respective upper bound on the approximation ratio. It turns out that these graphs have a small number of high degree nodes: The maximum degree is $\Theta(n)$, whereas the average degree is only $O(\sqrt{n})$. We wonder if these nodes of high degree are essential in order to enforce a bad performance for MINGREEDY? This question motivates an investigation of graphs whose maximum degree Δ is bounded above by a constant. For graphs with maximum degree at most three we show that MINGREEDY achieves an approximation ratio of $\frac{2}{3}$ (see Sect. 2.2). Moreover, we show that MINGREEDY gives a $\frac{\Delta-1}{2\Delta-3}$ -approximation on Δ -regular graphs for any Δ . We conjecture this guarantee to hold for all graphs with maximum degree Δ ; what we show now is a slightly weaker guarantee of $\frac{\Delta-1/2}{2\Delta-2}$.

Note that our guarantees even hold if all ties are broken deterministically, i.e. we select some minimum degree node arbitrarily and match it to an arbitrary neighbor. Therefore, it is even more striking that the worst case guarantee for MINGREEDY is better than the best known guarantee for the *expected* approximation ratio of 0.523 for RANKING [9] if $\Delta \leq 11$; compared to the best known bound for the expected approximation ratio of 0.5000025 [2] for MRG our guarantee is better whenever $\Delta \leq 100,001$ holds. Regarding GREEDY, our bound is better for all Δ than the known expected performance guarantee of $\frac{1}{2}(\sqrt{(\Delta-1)^2+1}-\Delta+2)$ shown by Miller and Pritikin [36]; in particular, GREEDY achieves an expected 0.618-approximation for $\Delta = 3$.

We also study the inherent limitations of greedy algorithms for the maximum matching problem. While the above mentioned algorithms are certainly all “greedy” in a very natural sense, we need

a formal characterization of what constitutes a greedy algorithm for the problem at hand. To this end, we use the model of *priority algorithms* introduced by Borodin, Nielsen, and Rackoff [8]. This model has been applied successfully to a large range of problems [13, 6, 38, 7, 28], see [28] for a recent summary. Greedy algorithms explore the input myopically, making an irrevocable decision for each piece. In the vertex model proposed in [13, 6] each piece of the input, also referred to as *data item*, corresponds to a node and its neighbors. An *adaptive priority algorithm* submits an ordering π on all data items without looking at the input \mathcal{I} , and receives the first data item in \mathcal{I} according to π , say for node u . Then the algorithm either matches u to one of its neighbors or isolates u . If the algorithm is additionally required to be *greedy*, then it does not have the option to isolate u . Note that the deterministic MINGREEDY (that breaks all ties arbitrarily) can be implemented as a greedy adaptive priority algorithm.

We show that no greedy adaptive priority algorithm achieves an approximation ratio better than $\frac{\Delta-1}{2\Delta-3}$ on graphs with maximum degree $\Delta \geq 3$ (see Sect. 3.1). In particular, no such deterministic algorithm can guarantee an approximation ratio of $\frac{1}{2} + \varepsilon$ for any $\varepsilon > 0$, therefore providing evidence that randomness is essential for greedy algorithms in order to guarantee a non-trivial approximation. For k -uniform hypergraphs we show a tight $\frac{1}{k}$ -inapproximability bound (see Sect. 4).

For priority algorithms that are not required to be greedy we show an inapproximability bound of $\frac{2}{3}$ (see Sect. 3); it relies on a graph of maximum degree three, hence our performance guarantee for MINGREEDY for $\Delta = 3$ is tight. We also study randomized priority algorithms introduced in [1] and show that these do not achieve an expected approximation ratio better than $\frac{5}{6}$ (see Sect. 3). Therefore, these randomized greedy algorithms cannot achieve the performance of algorithms based on augmenting paths or algebraic methods. We point out that our class of randomized priority algorithms is very comprehensive; in particular, it contains all algorithms mentioned so far as well as the randomized algorithm of Karp and Sipser [31, 3]. Furthermore, our class subsumes the models of randomized algorithms proposed by Goel and Tripathi [22]. In particular, their models do not contain MINGREEDY.

1.1 The Maximum Cardinality Matching Problem

Let $G = (V, E)$ be an unweighted and undirected graph. A matching $M \subseteq E$ is a selection of edges of G such that no two edges in M share a node. If no further edge of $E \setminus M$ can be added to M without violating this condition, then M is called a *maximal* matching. If M has largest cardinality among all matchings in G , then M is a *maximum* matching. If M covers all nodes, we call it *perfect*.

An algorithm is called an α -approximation algorithm for the maximum (cardinality) matching problem, if for any graph it finds in polynomial time a matching whose size is at least α times the size of a maximum matching. A randomized α -approximation algorithm is a polynomial time algorithm that obtains a matching whose *expected* cardinality is at least α times the optimal solution.

Note that any maximal matching M has size at least half of the optimum. To see this, choose any maximum matching M^* and observe that each edge in M shares nodes with at most two edges in M^* . Thus, any algorithm that finds a maximal matching is a $\frac{1}{2}$ -approximation algorithm. In particular, the greedy algorithms we consider in this article will have this property.

Related Work outside the Greedy Paradigm. A large variety of algorithmic techniques have been applied to the maximum matching problem. We provide a brief outline.

Edmonds [16] proposed the famous blossom algorithm that finds a maximum matching in polynomial time by detecting augmenting paths. Gabow [19] showed how to implement this algorithm with running time $O(|V|^3)$. Micali and Vazirani gave an exact algorithm with running time $O(\sqrt{|V|}|E|)$, that iteratively finds short augmenting paths. However, its analysis and implementation are non-trivial (cp. [43, 20] and further references therein).

Goldberg and Karzanov [23] proposed an algorithm based on flow techniques with a running time of $O(\sqrt{|V|}|E| \log(n^2/m)/\log n)$; to the best of our knowledge this is the fastest exact algorithm for dense non-bipartite graphs.

Using algebraic methods, a maximum matching can be found in time $O(|V|^\omega)$ by the randomized algorithm of Mucha and Sankowski [37] (see also [21, 24]), where ω is the exponent of the best known matrix multiplication algorithm.

2 The MinGreedy Algorithm

Encouraged by its excellent performance on random graphs, we study the worst case performance of MINGREEDY and present a family of hard graphs. In particular, our graphs show that the expected approximation ratio of MINGREEDY is at most $\frac{1}{2} + \varepsilon$ whp. for any $\varepsilon > 0$.

Choose $a, b \in \mathbb{N}$, where b is even and $b \mid a$, and let $c := 2 \cdot \lceil \sqrt{a} \rceil$. The graph $G_{a,b}$ has $2a+c$ nodes partitioned as follows:

$$\begin{aligned} S_1 &= \{1, 2, \dots, a\} \\ S_2 &= \{a+1, a+2, \dots, 2a\}, \text{ and} \\ S_3 &= \{2a+1, 2a+2, \dots, 2a+c\}. \end{aligned}$$

Edges are as follows (see Fig. 1 for an example of the graph $G_{a,4}$):

- i. Each node in S_1 is connected to *every* node in S_3 .
- ii. Each node $i \in S_1$ is connected to node $a+i \in S_2$.
- iii. Each S_3 -node is connected to exactly one S_2 -node.
- iv. The crucial ingredient is: within S_2 we form disjoint cliques of size b .

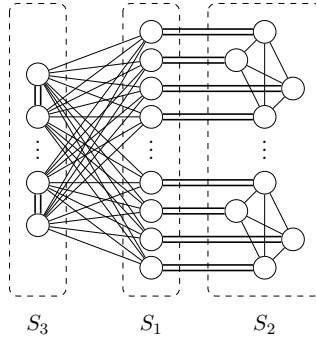


Figure 1: The graph $G_{a,4}$. The edges represented by double lines form a perfect matching

Observe that a perfect matching of size $a + \lceil \sqrt{a} \rceil$ consists of the a edges connecting the S_1 -node with the S_2 -nodes and the $\lceil \sqrt{a} \rceil$ edges within S_3 . The edges of the perfect matching are displayed as double edges in Fig. 1.

Our intention is that an instantiation of the MINGREEDY algorithm matches S_2 -nodes with S_2 -nodes initially, since these nodes will have minimum degree. Thereby the respective algorithm may add $\frac{|S_2|}{2} = \frac{a}{2}$ edges within the cliques of S_2 to its matching. When all S_2 nodes are matched, all remaining edges are incident with nodes in S_3 . Hence at most $|S_3| = 2\lceil \sqrt{a} \rceil$ additional edges can be added to the matching. If the randomized algorithm follows this intended scheme, then it achieves an approximation ratio of $(\frac{a}{2} + 2\lceil \sqrt{a} \rceil)/(a + \lceil \sqrt{a} \rceil)$ on the graph $G_{a,b}$, which tends to $\frac{1}{2}$ for large a . However, it turns out that the randomized MINGREEDY algorithm does not strictly adhere to this plan.

2.1 A Worst Case Analysis of MinGreedy and two Variants

Hougardy [27] considered deterministic algorithms that iteratively pick an edge (u, v) such that either u or v has currently minimum degree. He proposed a family of graphs for which any such algorithm achieves an approximation ratio of $\frac{1}{2} + o(1)$. Since the difficulty of these graphs relies on the assumption that all ties are broken towards the worst case, they are no obstacle for MINGREEDY.

We begin with a worst case study of Tinhofer's MINGREEDY. In each iteration the algorithm selects a node, the first endpoint, uniformly at random among all nodes of minimum degree and matches it to the second endpoint; this neighbor is also picked uniformly at random. It is a well-known fact that for every node v in a connected graph there is a maximum matching that covers v (see [17] for a stronger statement). This provides additional motivation for MINGREEDY, since we have a good chance to pick an optimal edge if the first endpoint has small degree. However, we show:

Theorem 1. *MINGREEDY cannot approximate the Maximum Matching Problem within $\frac{1}{2} + \varepsilon$ whp. (for any $\varepsilon > 0$), even on bipartite graphs.*

Proof. We consider the graph $G_{a,\sqrt{a}}$, and let a be a large (integral) square number for the sake of convenience. The crucial property P is that all nodes of minimal degree are in S_2 . Initially, this is truly the case: Recall that by construction every node in S_2 has degree \sqrt{a} , whereas the degrees are $2\sqrt{a} + 1$ in S_1 and $a + 1$ in S_3 .

In the following, the notion of a (S_j, S_k) *matching*, for $k, j \in \{1, 2, 3\}$, means that a node in set S_j is selected by the algorithm because of its minimal degree and matched to a neighbor in set S_k . When may the property P be violated?

- Every (S_2, S_2) matching reduces the degree of their clique-neighbors by 2 and the degrees of their *unique* S_1 neighbors by 1. Since any node in S_1 is affected only once, this case does not pose any danger.
- Every (S_2, S_1) matching reduces the degree of all nodes in S_3 and all nodes in the respective clique by 1, but it has no effect on any other node in S_1 . Hence, a large number of (S_2, S_1) matchings reduces the degree of nodes in S_3 from its initial value $a + 1$ below the degrees of nodes in S_2 (which is at most \sqrt{a}). In that event the algorithm prefers matching the nodes in S_3 , thereby drastically reducing the degrees in S_1 . As a consequence, a large number of nodes in S_1 might appear in the matching and the algorithm may come up with a good approximation. We will see, however, that this scenario is very unlikely.

First we assume that P is preserved. Later we show that P holds whp. throughout the computation. We need to bound the expected number of (S_2, S_1) matchings. Observe that the algorithm processes the cliques of S_2 one after another, fixing all nodes of a clique before moving on to the next.

At first we focus on a single clique in S_2 that initially has \sqrt{a} nodes. The probability that a node $v \in S_2$ is matched to its neighbor $u \in S_1$, conditioned on the event that MINGREEDY selects v as first endpoint, depends on the number x of (S_2, S_2) matchings and the number y of (S_2, S_1) matchings that have been performed in the respective clique so far. In particular, the conditional probability equals $\frac{1}{\deg(v)-2x-y} = \frac{1}{\sqrt{a}-2x-y}$, since v has $\sqrt{a}-1$ neighbors in S_2 and one in S_1 . Then the expected number of (S_2, S_1) matchings (for a single clique) is given by the sum of such probabilities, where $0 \leq 2x + y < \sqrt{a}$ holds.

For what values of x and y a random trial is performed depends on the outcome of previous trials. We attain a worst case perspective. Let Z_i for $0 \leq i < \sqrt{a}$ be a binary random variable with success probability $p_i = \frac{1}{\sqrt{a}-i}$. Then $\sum_i Z_i$ is an upper bound on the number of (S_2, S_1) matchings for a single clique. If a is sufficiently large, the expected value is upper-bounded by $\mathbb{E}[\sum_i Z_i] = \sum_{i=1}^{\sqrt{a}} \frac{1}{i} \leq \ln(a)$, where it is implicitly assumed that all previous actions were (S_2, S_1) matchings.

To show that property P is likely to be preserved, we show that $\sum_i Z_i$ is concentrated at its expected value, using a variant of the Chernoff bound (cp. Sect. 1.6 in [14]). Then,

$$\text{prob} \left[\sum_i Z_i \geq (1 + \beta) \cdot \ln(a) \right] \leq \left(\frac{e^\beta}{(1 + \beta)^{(1+\beta)}} \right)^{\ln(a)} \leq \frac{1}{a^{\frac{3}{2}}}$$

where the second inequality follows by choosing $\beta = e^2$. We employ the Union bound to obtain a lower bound of $1 - \sqrt{a}/a^{\frac{3}{2}} = 1 - \frac{1}{a}$ on the probability that for none of the \sqrt{a} cliques inside S_2 the sum deviates by more than a factor of $1 + \beta$ from its expected value.

Hence, there are whp. at most $O(\sqrt{a} \cdot \ln(a)) = o(a)$ (S_2, S_1) matchings in total and in particular property P is preserved whp. throughout the course of the algorithm, since a violation would require $\Omega(a)$ (S_2, S_1) matchings. Furthermore, the size of the matching returned by MINGREEDY on $G_{a, \sqrt{a}}$ is at most $\frac{a}{2} + o(a)$ with high probability, whereas the maximum matching in the graph has cardinality at least a .

To obtain a bipartite graph, we take two copies L, R of $G_{a, \sqrt{a}}$ and let $S_i := S_i^L \cup S_i^R$. First we remove the edges within S_3 and create an edge for each node in S_3^L and its counterpart in S_3^R . Then we remove all edges within S_2 and form complete bipartite cliques of each former clique in S_2^L and its counterpart in S_2^R . Note that the degree of nodes in S_2 have increased by one, which is negligible in our analysis. All other degrees do not change. The maximum matching increased by at least a edges, whereas the output matching grows by $\frac{a}{2} + o(a)$. \square

There are two natural variants of MINGREEDY. The first variant, EDSM (enhanced Degree-Sequenced Matching), matches a node of minimum degree to a neighbor that has in turn minimum degree among all neighbors. All ties are broken uniformly at random. Hosaagrahara and Sethu [26] apply EDSM to the problem of assigning network packets from input ports to output ports to achieve low average delay and find that it performs very well on real and synthetic traffic traces. Another variant proposed in [26] reduces to EDSM on our hard instances. MDS (Minimum Degree Sum) picks the next edge such that the sum of the degrees of its endpoints is minimal, also breaking ties at random. Both algorithms ignore edges incident with already matched nodes.

EDSM and MDS achieve only a trivial expected approximation ratio, as we show next. For MDS, this fact also follows from the definition of graphs given in [27]. We present a unified construction that applies to both algorithms.

Theorem 2. *For any $\varepsilon > 0$, there is a bipartite graph such that neither EDSM nor MDS achieve approximation ratio $\frac{1}{2} + \varepsilon$.*

Proof. As worst case input we use the graph $G_{a,2}$. See the beginning of Sect. 2 for description of the family of graphs, and our intention of how the algorithm should proceed. We show below how to make the graph bipartite. Recall that the degrees of S_3 - and S_1 -nodes increase with a , whereas all S_2 -nodes have degree two.

As desired, both EDSM and MDS start by repeatedly matching a node in S_2 with its unique neighbor in S_2 until all nodes in S_2 are matched. Thus, the claimed bound on the approximation ratio follows from the discussion at the beginning of Sect. 2.

To obtain a bipartite graph, we first remove all inner edges of S_3 , thereby decreasing the size of the maximum matching by $\lceil \sqrt{a} \rceil$. Note that if all nodes of S_1 were connected to all nodes in S_3 , there would be cycles of odd length. Hence we add a set S'_3 of nodes, with $|S'_3| = |S_3| = 2\lceil \sqrt{a} \rceil$, and connect odd S_1 -nodes with S_3 -nodes and even S_1 -nodes with S'_3 -nodes such that these edges are evenly distributed over the nodes of S_3 and S'_3 . The matching obtained by the algorithm is increased by $o(a)$ only. \square

2.2 Guarantees for MinGreedy on Bounded Degree Graphs

In the construction of Theorem 1 the approximation ratio of MINGREEDY converges to $\frac{1}{2}$ as the maximum degree increases. In this section we study degree bounded graphs.

We state that if all degrees are bounded above by $\Delta = 2$, then any algorithm that picks an edge incident to a node of degree one (if such a node exists) computes a maximum matching. Thus, in particular MINGREEDY and KARPSIPSE [31], that picks a random edge incident with a degree-1 node, if such an edge exists, and a random edge otherwise, are optimal on such graphs.

In what follows we show that if the degrees in the input are bounded above by $\Delta \geq 3$, then the approximation ratio of MINGREEDY is strictly better than $\frac{1}{2}$. Interestingly, our guarantee holds for any algorithm that iteratively picks an edge that is incident with some node of current minimum degree. Thus, we do not use the feature that MINGREEDY breaks all ties uniformly at random. In particular, our bounds hold for the deterministic variant of MINGREEDY that picks an arbitrary non-isolated node of minimum degree and matches it with an arbitrary neighbor. Then both nodes and all their incident edges are removed from the graph. Therefore each node left in the graph is unmatched, and the algorithm iterates on the remaining vertices.

First we study graphs of bounded degree 3 and show that the deterministic variant of MINGREEDY, achieves an approximation ratio of $\frac{2}{3}$. Along the same lines, we also obtain a guarantee of $\frac{\Delta-1}{2\Delta-3}$ for Δ -regular graphs.

In Sect. 2.3 we prove a slightly worse bound for graphs of maximum degree Δ .

Theorem 3. *If all degrees are bounded above by $\Delta = 3$ or the input graph is Δ -regular with $\Delta \geq 4$, then MINGREEDY achieves approximation ratio at least $\frac{\Delta-1}{2\Delta-3}$.*

More generally, these bounds hold for any algorithm which selects edges incident with a node of minimum degree.

Let $G = (V, E)$ be a connected graph and M the matching computed by MINGREEDY. Given M we choose a maximum matching M^* for the analysis such that the connected components of the graph $H = (M \cup M^*)$ are of two types: An *augmenting path* X has $m_X \geq 1$ edges of M and $m_X^* = m_X + 1$ edges of M^* that alternate. In particular, any augmenting path starts and ends with an M^* -edge. The second type are *singletons*. Such a component is a path of length one in H , i.e. its edge belongs both to M and M^* .

Why can we always find such an optimal matching M^* ? Observe that H has maximum degree two, hence it consists of cycles and paths. Every component that is not a singleton alternates between M and M^* , since every node has degree at most one in each matching. Then the crucial observation is that each component X that is either an even-length cycle or an even-length path can be replaced by m_X singletons; we simply exchange the respective edges of M^* by the ones of M .

Local Approximation Ratios. To obtain a bound of $\alpha = |M|/|M^*| \geq \frac{\Delta-1}{2\Delta-3}$ on the global approximation ratio α of MINGREEDY we want to bound *local approximation ratios*

$$\alpha_X = m_X / m_X^*$$

of components X of H . We call an augmenting path of length three a $\frac{1}{2}$ -path. A $\frac{1}{2}$ -path X has local approximation ratio only $\alpha_X = \frac{1}{2} < \frac{\Delta-1}{2\Delta-3}$, whereas longer augmenting paths X have $\alpha_X \geq \frac{2}{3} \geq \frac{\Delta-1}{2\Delta-3}$ since $\Delta \geq 3$. A singleton X even has $\alpha_X = 1$. Therefore we balance local approximation ratios. We say that a component X has M -funds m_X and introduce a change t_X to the M -funds of X such that

$$\alpha_X = \frac{m_X + t_X}{m_X^*} \stackrel{!}{\geq} \frac{\Delta - 1}{2\Delta - 3}$$

holds for the new local approximation ratio of X . By transferring M -funds between components we assert that $\sum_X t_X = 0$ holds. Then the total M -funds $\sum_X m_X + t_X = \sum_X m_X = |M|$ are unchanged and MINGREEDY achieves approximation ratio at least

$$\alpha = |M|/|M^*| = \left(\sum_X m_X + t_X \right) / |M^*| \geq \left(\sum_X \frac{\Delta - 1}{2\Delta - 3} \cdot m_X^* \right) / |M^*| = \frac{\Delta - 1}{2\Delta - 3}.$$

The Choice of Transfers. Our approach is to transfer M -funds between components using a selection of edges in $F := E \setminus (M \cup M^*)$. We develop a charging scheme that transfers M -funds from M -covered nodes to adjacent endpoints of augmenting paths. Note that F does not contain edges between endpoints of augmenting paths, since otherwise M would not be maximal.

In particular, we will assert that components whose local approximation ratio is insufficient receive the required M -funds. These components in need are $\frac{1}{2}$ -paths, i.e. augmenting paths of length three.

First we show that every endpoint of an augmenting path is incident to at least one edge in F . Let $d_G(w)$ denote the degree of node w in G .

Lemma 1. *Let w be an endpoint of an augmenting path. Then $d_G(w) \geq 2$ holds.*

Proof. Let X be the augmenting path of w . Consider the step when MINGREEDY picks the first M -edge e in X . Edge e and both its adjacent M^* -edges must still be contained in the graph. Since MINGREEDY selects a minimum degree node, all nodes of X have degree at least two. \square

Which edges in F are selected to transfer funds?

Definition 1. Let (v, w) be an edge in F , where v is covered by M and w is an endpoint of an augmenting path. Assume that v is matched in step s . Then edge (v, w) is a transfer if at the end of step s the degree of w is $d(w) \leq \Delta - 2$.

We frequently denote a transfer as vw to stress its direction from the M -covered node v to the augmenting path endpoint w . In order to refer to transfers to and from a given component X , we also call vw a *credit* to w respectively a *debit* to v . Next we show that each endpoint of an augmenting path is guaranteed to receive transfers.

Lemma 2. Let degrees be bounded by Δ . The number of credits c_w to an augmenting path endpoint w is at least $c_w \geq \min\{d_G(w) - 1, \Delta - 2\} \geq 1$.

Proof. By Lemma 1 we have $d_G(w) \geq 2$, thus $\min\{d_G(w) - 1, \Delta - 2\} \geq 1$ holds.

If $d_G(w) \leq \Delta - 1$ holds, then any time the degree of w drops, it drops to at most $\Delta - 2$. Therefore all F -edges incident with w are credits.

If $d_G(w) = \Delta$ holds, then after $d(w)$ drops to $d(w) = \Delta - 1$, all F -edges of w removed later are credits: at most one F -edge of w is not a credit. \square

Bounding Local Approximation Ratios. Let d_X (resp., c_X) denote the numbers of debits (respectively credits) to the nodes of a component X . We call $d_X - c_X$ the *balance* of X . Our approach is to move a constant amount θ of M -funds along each transfer. Assuming that $d_X - c_X \leq T_X$ holds, the local approximation ratio of X is at least

$$\alpha_X = \frac{m_X - \theta d_X + \theta c_X}{m_X^*} \geq \frac{m_X - \theta T_X}{m_X^*}.$$

To prove Theorem 3 we find θ and T_X such that $\alpha_X \geq \frac{\Delta-1}{2\Delta-3}$ holds for all X .

The maximum *possible* number of debits to a component depends on its number of M -covered nodes and on Δ : By the degree constraint, each M -covered node of X has at most Δ debits. However, no funds are moved over edges in $M \cup M^*$, and if X is an augmenting path, then its path endpoints do not incur any debits at all. We say that a component has *two missing debits*, if its actual number of debits is at least two less than the maximum possible number.

First consider a singleton X . By definition, X does not get credits, i.e. $c_X = 0$. By the above considerations, both nodes of X have at most $\Delta - 1$ debits each. Using two missing debits, the balance of X and the local approximation ratio of X are

$$d_X - c_X = d_X \leq 2(\Delta - 1) - 2 \tag{1}$$

$$\alpha_X = \frac{1 - \theta(d_X - c_X)}{1} \geq 1 - 2\theta(\Delta - 2). \tag{1'}$$

Note that $\alpha_X \geq \frac{\Delta-1}{2\Delta-3}$ holds if we choose $\theta \leq \frac{1}{2(2\Delta-3)}$.

Let X be an augmenting path. Since degrees are at most $\Delta = 3$ or the graph is Δ -regular, by Lemma 2 credits to a path endpoint w of X (respectively in total to both path endpoints of X) are

$$c_w \geq \Delta - 2 \quad \text{respectively} \quad c_X \geq 2 \cdot (\Delta - 2). \tag{2}$$

From the above considerations $d_X \leq 2m_X(\Delta - 2)$ follows. Assuming that two debits are missing, we obtain

$$d_X - c_X \leq 2m_X(\Delta - 2) - 2(\Delta - 2) - 2 \quad (3)$$

$$\begin{aligned} \alpha_X &= \frac{m_X - \theta(d_X - c_X)}{m_X^*} \geq \frac{m_X - 2\theta m_X(\Delta - 2) + 2\theta(\Delta - 1)}{m_X + 1} \\ &= 1 - 2\theta(\Delta - 2) + \frac{2\theta(2\Delta - 3) - 1}{m_X + 1} = 1 - 2\theta(\Delta - 2), \end{aligned} \quad (3')$$

where we choose $\theta = \frac{1}{2(2\Delta-3)}$ in the last equality.

Combining Eq. (1') and Eq. (3') yields our claimed performance guarantee for MINGREEDY: the local approximation ratio of each component X is $\alpha_X \geq 1 - 2\theta(\Delta - 2) = 1 - \frac{2(\Delta-2)}{2(2\Delta-3)} = \frac{\Delta-1}{2\Delta-3}$.

The Proof of Theorem 3 for Δ -Regular Graphs with $\Delta \geq 4$

First we show that each component X has two missing debits, be it due to a node in X having low degree in G or being incident to non-transfer F -edges. The following steps of MINGREEDY are crucial: We say that a node u *creates* its component X , if MINGREEDY selects node u as the first endpoint of the first edge matched in X .

Lemma 3. *Let u create X and $d(u)$ be its degree at creation. X has two missing debits if*
a) $d(u) \leq \Delta - 2$, b) $d(u) = \Delta$, c) $\Delta \geq 4$, or d) X is a singleton.

Proof. In order to obtain a contradiction, we assume that at most one debit to a node of X is missing. Let X be created in step s when u is selected because of its minimum degree $d(u)$ and matched to v . By assumption, one of u, v has a debit.

If $d(u) \leq \Delta - 2$ at step s , then u has two missing debits by Definition 1. This proves part a), and we may assume $d(u) \geq \Delta - 1$ from now on. Here is an overview of the argument. Consider a debit xw' to $x \in \{u, v\}$. By Definition 1, step s removes edge (x, w') from G and, by definition of transfers, after step s the degree of w' is $d'(w') \leq \Delta - 2$. In particular, after step s the degree of w' is smaller than the degree of u before step s . Our crucial claim is:

At step $s+1$ there is an augmenting path endpoint w that currently has minimum degree $d'(w)$ with $\Delta - 2 \geq d'(w) \geq 1$. In particular, w is not isolated.

Note that node w has a neighbor in X , but does not necessarily belong to X . Since w is an augmenting path endpoint, it cannot be the node selected by MINGREEDY in step $s + 1$. Hence it cannot be the case that at step $s + 1$ all minimum degree nodes are augmenting path endpoints: There must be a node $y \neq w$ that is selected in step $s+1$ with minimum degree $d'(y) = d'(w) \leq \Delta - 2$. Since the degree of y was at least $d(u) = \Delta - 1$ before step s , in step s edges of F connecting y with u or v are removed. But u, v, y are M -covered, hence the removed edges are not debits to u, v . We obtain a contradiction if we find at least two missing debits.

b) Assume that $d(u) = \Delta$ holds at step s . Since in step s at most two edges incident with w are removed and there is a transfer xw with $x \in \{u, v\}$, we have $d'(w) = \Delta - 2$ at step $s + 1$. In particular, our claim holds: node w is not isolated, since $d'(w) = \Delta - 2 \geq 1$, and w has minimum degree. The reason is that every node had degree Δ before step s , and every degree drops at most by two. So let $y \neq w$ be the node selected in step $s + 1$. The degree of y also drops from Δ

to $\Delta - 2$ in step s when incident edges $(u, y), (v, y)$ are removed. If y is not a node of X , then both $(u, y), (v, y)$ are F -edges but they are not debits, since u, v, y are M -covered. Hence we have found for each of u, v a missing debit. If y is a node of X , then u, v, y form a triangle and one of $(u, y), (v, y)$, say (u, y) , is an F -edge. Since both u, y are M -covered, edge (u, y) is not a transfer and hence both u, y have a missing debit.

c) Since parts a) and b) apply to $\Delta \geq 4$, we may assume that $d(u) = \Delta - 1$ holds at step s . Recall that node u has a missing debit due to its low degree. We study step $s + 1$ to find an additional missing debit. Our claim holds: since xw is a transfer, after step s the degree $d'(w) \leq \Delta - 2$ is smaller than that of u before step s ; using $\Delta \geq 4$ we get $d'(w) \geq \Delta - 3 \geq 1$, since at most two edges incident with w are removed in step s . Let $y \neq w$ be the neighbor of u or v being selected with degree $d'(y) \leq \Delta - 2$ next. Recall that step s removes an edge (x, y) with $x \in \{u, v\}$. If y is not a node of X , then (x, y) is an F -edge. Since x, y are M -covered, edge (x, y) is not a transfer and we have found the other missing debit to one of u, v . Now assume that y is a node of X . At step $s + 1$ node y is incident with its M -edge, maybe with debits and possibly with its M^* -edge. Observe that no matter if the M^* -edge of y is already removed, node y has a missing debit since $d(y) \leq \Delta - 2$. We have found the second missing debit and obtain a contradiction to the assumption that X has at most one missing debit.

d) By parts a)-c) it suffices to prove the case that degrees are bounded by $\Delta = 3$ and $d(u) = \Delta - 1 = 2$ holds. For singleton X both endpoints can have at most three debits, since the edge of X belongs to M and hence does not move funds. Since u has a missing debit and by our assumption at most one debit is missing for X , we get that u has exactly one debit, say to w_u . Then v has exactly two debits, say to w_v and w'_v .

Thus, u can be adjacent to at most one of w_v and w'_v , since u has degree two and is already adjacent to v . Since the edges (u, w_u) , (v, w_v) , and (v, w'_v) are transfers, Definition 1 implies that the degrees of w_u , w_v and, w'_v are each at most $\Delta - 2 = 1$ after u was matched to v .

If u is adjacent to neither w_v nor w'_v , then w_u , w_v , and w'_v have degree one afterwards, since their degrees were at least the minimum degree of $d(u) = 2$ before (u, v) was matched and dropped to at most one afterwards. Now consider the case that w_u is also a neighbor of v , say $w_v = w_u$. Then the degree of w'_v drops by at most one when the edge (u, v) is picked by the algorithm, and hence w'_v has degree one afterwards.

In both cases w_u, w_v or w'_v must be selected and matched by the algorithm in step $s + 1$, since no other degrees dropped in step s . A contradiction: w_u, w_v, w'_v are assumed to be augmenting path endpoints. \square

We prove Theorem 3 for Δ -regular graphs with $\Delta \geq 4$. A singleton X has two missing debits by Lemma 3d): the balance is at most $d_X - c_X = d_X \leq 2(\Delta - 1) - 2$ as required in Eq. (1). An augmenting path X has two missing debits by Lemma 3c) and $c_X \geq 2(\Delta - 2)$ credits by Eq. (2): the balance is at most $d_X - c_X \leq 2m_X(\Delta - 2) - 2 - 2(\Delta - 2)$ as claimed in Eq. (3).

The Proof of Theorem 3 for Graphs of Degree At Most $\Delta = 3$

By Lemma 3d), a singleton X has a balance of at most $d_X - c_X = d_X \leq 2(\Delta - 1) - 2 = 2$ as required in Eq. (1). We prove Eq. (3) for an augmenting path X . By Eq. (2), each path endpoint of X receives at least one credit. If X has two missing debits, then the balance of X is at most $d_X - c_X \leq 2m_X - 4$, i.e. X receives a sufficient amount of M -funds. Hence we assume from now

on that X has at most one missing debit and that each path endpoint of X receives exactly one credit.

Lemma 3a) and b) imply that X has two missing debits if the degree of u in the creation step of X is *not* $\Delta - 1 = 2$. Thus, we focus on the case that the degree of u is $d(u) = 2$ at creation. Since u is incident with an M -edge and an M^* -edge, it is not incident with an F -edge. Therefore, u has a missing debit. According to our assumption this is the only missing debit of X .

Lemma 4. *Assume that the graph has maximum degree $\Delta = 3$. If X is an augmenting path with $d_X = 2m_X - 1$ debits, then at least $c_X \geq 3$ credits are given to X .*

Proof. To show the statement we assume that $c_X < 3$ holds and show a contradiction.

We have already argued that each path endpoint of X receives exactly one credit, that the node u selected to create X has degree $d(u) = 2$ at creation, and that u has the only missing debit of X . Then all other M -covered nodes of X have exactly one debit in order to ensure $d_X = 2m_X - 1$. Assume that v is matched with u in M .

First we consider the case of $m_X = 1$, i.e. that X is a $\frac{1}{2}$ -path. Let w_u, w_v be the endpoints of X such that $(u, w_u), (v, w_v) \in M^*$. Note that this implies $w_u \neq w_v$. Then v has the only debit, say to the augmenting path endpoint z_v . When u and v are matched, the degrees of w_u, w_v, z_v all drop, and no other degrees drop. In particular, we claim that w_v has degree exactly one afterwards, which is the new minimum degree in the graph. As a consequence, one of w_v, w_u, z_v is matched in the next round. A contradiction is obtained since augmenting path endpoints are never matched.

Now we prove the claim. When the component is created $d(u) = 2$ holds. Since u is adjacent to v and w_u , it is not a neighbor of node w_v . Hence w_v is incident with at least one F -edge. Since only the M^* -edge of w_v is removed, w_v is still incident with at least one F -edge after creation. If w_v is still incident with at least two F -edges, then w_v receives at least two credits, i.e. at least one more than assumed. So after creation w_v is incident with exactly one F -edge and has degree one.

Next consider the case that the edge (u, v) belongs to an augmenting path with $m_X \geq 2$ edges in M . Since $(u, v) \in M$ is the first edge picked by the algorithm in X , at least one path endpoint w of X is still connected with its unique neighbor x in M^* , after u and v have been removed with all their incident edges. We consider the step when x is matched to its neighbor in M , say x' . Node w and the recipient of the debit to x are not yet isolated, since these two nodes are never matched. Thus x has degree three before this step.

Moreover, we may assume that w has degree at most two before this step. To see this, assume that w 's degree were still $\Delta = 3$ and observe that the two F -edges incident with w would become credits, i.e. node w would get more credits than assumed.

Since x' has exactly one debit, i.e. x' is adjacent to one endpoint of an augmenting path, node x' has degree exactly two before the step. Consequently, node w has degree exactly two as well.

All still present neighbors of x, x' are endpoints of augmenting paths, and only their degrees drop when x and x' are removed from the graph. If x' is not adjacent to w , then w 's degree drops by one and to exactly one, hence either w or another augmenting path endpoint would be picked in the subsequent step. A contradiction, since an augmenting path endpoint is never matched.

Lastly, assume that x' and w are adjacent; then w becomes isolated in that step. We consider the recipient y of the debit to x . Since x' is adjacent to w and x and has degree exactly two, node y is not adjacent to x' . Therefore the degree of y drops by exactly one. Since y 's degree drops from at least two before the step to at most $\Delta - 2 = 1$ afterwards by Def. 1 we get that y 's degree is exactly one in the subsequent step. Furthermore, node y is now the only degree-1 node and is matched next. A contradiction since y is an augmenting path endpoint. \square

2.3 A Guarantee for Graphs with Maximum Degree $\Delta \geq 4$

In this section we consider graphs with maximum degree $\Delta \geq 4$ and show a slightly weaker bound. As in Theorem 3, our guarantee holds for a more general class of greedy algorithms that repeatedly add an edge (u, v) to the matching such that either u or v has current minimum degree. For instance, this holds for the deterministic variant of MINGREEDY that breaks all ties arbitrarily. We show the following guarantee.

Theorem 4. *MINGREEDY achieves an approximation ratio of at least $\frac{\Delta-1/2}{2\Delta-2}$ on graphs with degrees at most Δ .*

Why does our bound not match our conjectured approximation ratio of $\frac{\Delta-1}{2\Delta-3}$?

In Theorem 3 we proved a guarantee of $\frac{\Delta-1}{2\Delta-3}$ for Δ -regular graphs. In order to provide sufficient M -funds to $\frac{1}{2}$ -paths, we used that each endpoint of such a short augmenting path receives the required number of incoming transfers. This was possible, since in a Δ -regular graph each endpoint has $\Delta - 1$ neighbors besides its mate in M^* .

This property no longer holds for the more general class of graphs we consider now. Thus, we modify our system of transfers.

Recall that an edge in M creates a component X in $H = (V, M \cup M^*)$ if that edge is the first edge that the algorithm picks in X .

In our new system nodes of an M -edge (u, v) that creates some augmenting path do not transfer M -funds, i.e. the F -edges of u and v are not debits. As a consequence, a $\frac{1}{2}$ -path does not have any debits, because it contains exactly one edge of M . Other than that we do not change the definition of transfers. Note that singletons are not augmenting paths and therefore might have debits.

In Lemma 1 we have shown that each endpoint of an $\frac{1}{2}$ -path has at least one F -edge, i.e. one edge that is eligible for a transfer. Indeed, as we show next, every $\frac{1}{2}$ -path receives at least one credit. We prove the following statement for augmenting paths of arbitrary lengths.

Lemma 5. *Let X be an augmenting path. Then at least one of its endpoints is not isolated immediately in the step when the first M -edge of X is added to the matching.*

Let w be an endpoint of X that is not immediately isolated. If w is not incident with its M^ -edge when it is eventually isolated in step s , then w receives a credit along one of the edges that it was incident with at the beginning of step s .*

Proof. If X contains more than one edge of M , then at most one M^* -edge can be adjacent to the M -edge picked first, and the first part of the lemma follows. Thus, assume that X is a $\frac{1}{2}$ -path. Assume that both endpoints of X become isolated upon removing the M -edge (u, v) , and note that u and v both have degree at least two. Then the endpoints must also have degree two and thus be adjacent to both u, v ; recall that the algorithm picks a node of minimum degree. But then the degree of u and v is in fact three. This implies that the endpoints must also have degree three, and hence they cannot be isolated by removing u and v only.

Now we proof the second part of the lemma. By assumption of the lemma, at the beginning of step s the incident edges of endpoint w are F -edges, and since w is assumed to be isolated in this step, there can be at most two. Assume that u' is matched to v' in step s , and recall that their F -edges provide credit for the endpoint w unless the algorithm creates a new augmenting path by picking the M -edge (u', v') . Thus, it suffices to show that picking (u', v') does not create an augmenting path. Since w has degree at most two, u' also has degree at most two. If $d(u') = 1$,

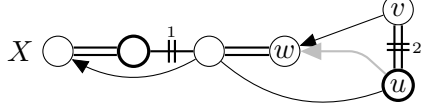


Figure 2: A $\frac{1}{2}$ -path X with only one credit from another component: M -edges are picked in order indicated by small numbers (from left to right) selecting fat nodes, the gray transfer is *not* an edge of G and does not effect computations of MINGREEDY

then u' does not have both an incident M -edge and M^* -edge and hence step s does not create an augmenting path. Since w becomes isolated but is not connected to u' , then v' is connected to w by exactly F -edge that provides a credit to w .

On the other hand, if $d(u') = 2$, then $d(w) = 2$. Since w becomes isolated, it must be adjacent to u' and v' . If u' belongs to X , then picking u' does not create X , since X was created earlier. Hence u' and v' both provide internal credit to w . Else u' is not a node of X . Since u' is connected to w , node u' is not incident both to an M -edge and an M^* -edge. But then matching u' and v' does not create a new augmenting path. Again both u', v' transfer a credit to w . \square

We briefly note that the first part of the lemma also implies that the approximation ratio of MINGREEDY may converge to $\frac{1}{2}$ asymptotically as shown in Theorem 1, but will never attain that value exactly.

Since we ensured that a $\frac{1}{2}$ -path X has no debits, we claim that the local approximation would be sufficient if we could always provide a second credit. However, it might be the case that the endpoints of X have only one neighbor outside X in total (cf. Fig. 2 for an example). In this case the second credit to X will be given via an *indirect transfer*; this is an M -fund that is not transferred via an F -edge (see the gray transfer in Fig. 2). To distinguish indirect transfers from F -edges which move M -funds, we also call the latter *direct transfers* from here on.

Interestingly, indirect transfers are also issued by components that are connected to X via an F -edge. Also, each indirect transfer originates at a node which is selected when it has degree one. Consequently, no indirect transfer leaves X . So, as desired, no direct or indirect transfers leave X at all.

We formally define indirect transfers in Sect. 2.3.2. But first we argue that a second incoming transfer to X is indeed sufficient to guarantee the claimed local approximations.

2.3.1 Optimizing Transferred M -Funds

Our upper bounds on the balance in Eq. (1) and Eq. (3) for singletons respectively augmenting paths are no longer valid for our new system of transfers. In the sequel we show upper bounds are exactly larger by one unit: A singleton X can have a balance up to

$$d_X - c_X = d_X \leq 2(\Delta - 1) - 2 + 1 \quad (4)$$

and the balance of an augmenting path X with $m_X \geq 2$ is bounded by at most

$$d_X - c_X \leq 2m_X(\Delta - 2) - 2(\Delta - 2) - 2 + 1. \quad (5)$$

We have already argued that a $\frac{1}{2}$ -path X has a sufficient balance if we can provide credits over two transfers, since no transfers leave X :

$$d_X - c_X \leq 0 - 2 = -2. \quad (6)$$

Assume we would move the amount $\theta = \frac{1}{2(2\Delta-3)}$ of M -funds along each transfer, as we did in case of Δ -regular graphs. Then a singleton with maximum balance $d_X = 2(\Delta - 2) + 1$ would have a local approximation ratio of only

$$\alpha_X = \frac{1 - \theta(d_X - c_X)}{1} = 1 - \frac{2(\Delta - 2) + 1}{2(2\Delta - 3)} = \frac{1}{2}.$$

In order to obtain a sufficient local approximation for all components, we adjust θ . What θ should we choose? Parametrized by θ , we lower bound local approximation ratios for all components and then optimize θ .

$$\frac{1}{2}\text{-paths:} \quad \alpha_X = \frac{1 - \theta(d_X - c_X)}{2} \stackrel{(6)}{\geq} \frac{1 + 2\theta}{2} = \frac{1}{2} + \theta \quad (5')$$

$$\text{singletons:} \quad \alpha_X = \frac{1 - \theta(d_X - c_X)}{1} \stackrel{(4)}{\geq} 1 - 2\theta(\Delta - 2) - \theta \quad (6')$$

$$\begin{aligned} \text{augmenting paths:} \quad \alpha_X &= \frac{m_X - \theta(d_X - c_X)}{m_X + 1} \\ &\stackrel{(5)}{\geq} \frac{m_X - \theta(1 + 2m_X(\Delta - 2) - 2(\Delta - 1))}{m_X + 1} \\ &= \frac{m_X(1 - 2\theta(\Delta - 2)) - \theta \cdot (1 - 2(\Delta - 1))}{m_X + 1} \\ &= 1 - 2\theta(\Delta - 2) + \frac{2\theta(2\Delta - 3) - 1 - \theta}{m_X + 1} \end{aligned}$$

Recall that in the last bound for augmenting paths we have $m_X \geq 2$. So if we choose $\theta \geq \frac{1}{2(2\Delta-2)}$, then $2\theta(2\Delta - 3) - 1 \geq -2\theta$ holds and we can simplify the bound to

$$\alpha_X \geq 1 - 2\theta(\Delta - 2) - \frac{3\theta}{3} = 1 - 2\theta(\Delta - 2) - \theta. \quad (7')$$

Combining Eq. (5'), Eq. (6'), and Eq. (7'), we set $\frac{1}{2} + \theta = 1 - 2\theta(\Delta - 2) - \theta$ and obtain $\theta = \frac{1}{2(2\Delta-2)}$. Hence the local approximation ratio of any component X is lower bounded by

$$\alpha_X \geq \frac{1}{2} + \theta = \frac{1}{2} + \frac{1}{2(2\Delta - 2)} = \frac{2\Delta - 1}{2(2\Delta - 2)} = \frac{\Delta - 1/2}{2\Delta - 2},$$

which proves our claimed performance guarantee for MINGREEDY.

2.3.2 Indirect Transfers

To show the balance bounds claimed for singletons and augmenting paths in Eq. (4) respectively Eq. (5), we first have to develop the definition of indirect transfers. Therefore we examine the properties of a $\frac{1}{2}$ -path for which an incoming indirect transfer needs to be added.

Let w be the endpoint of X that has receives the only direct credit. Denote by w' the other endpoint of X and assume that X is created in step s .

- (c1) Node w' must be isolated in step s . Otherwise w' would also receive a direct credit by Lemma 5 since it loses its M^* -edge in step s .
- (c2) Node w is not isolated in step s , also by Lemma 5.
- (c3) Let step $s' > s$ be the step in which w becomes isolated. At this time w has exactly one neighbor and receives exactly one direct credit. To see this, assume that w would be adjacent with both nodes removed in step s' . Then each such F -edge would provide one direct credit, since w 's degree drops from two to zero. But by assumption w receives only one direct credit.

What are the steps leading to w becoming isolated eventually? Recall that an F -edge does *not* provide a direct transfer to w if the respective neighbor is removed upon creation of a new augmenting path. Thus:

- (c4) If w has degree larger one after its own $\frac{1}{2}$ -path is created, then until its degree reaches one it only drops in steps when a new augmenting path is created (and one or both of the nodes matched first is adjacent to w).

Why? If an adjacent node z of w is matched and z would not belong to the first M -edge of an augmenting path, then the edge (z, w) would provide a second direct credit to w . But we assumed that w only receives one direct credit.

Assume that in step s' the algorithm selects node u and matches it to v . Note by (c3) node w has degree one in that step, hence the degree of u is also one. Moreover, v is w 's last neighbor, since w is isolated by removing u and v . Therefore, (v, w) is a direct transfer.

- (c5) Since u has degree one when it is matched, all other neighbors (if any) must be matched. Thus, u has no direct debits to its neighbors.

Since u has no direct debits on its own, we add an indirect transfer uw from u to v . Moreover, vw is the direct transfer to X .

Definition 2. Let X be a $\frac{1}{2}$ -path that receives exactly one direct credit vw to a path endpoint w of X . Let u be the mate of v in M . Then we add a transfer uw and call it an indirect transfer.

In particular, given X the two nodes that participate in the indirect transfer are uniquely defined. However, it might be that several $\frac{1}{2}$ -paths with exactly one direct credit require an indirect transfer from the same node. Finally we remark:

- (c6) No indirect transfer leaves node v , since by construction indirect transfers are added only for its unique M -neighbor u .
- (c7) Recall that in our new system of transfers the first M -edge (x, x') picked in an augmenting path does not have any outgoing direct or indirect transfers.

2.3.3 The Upper Bound on the Balance of Singletons

For a singleton $X = (u, v)$ we claimed in Eq. (4) a balance of at most

$$d_X - c_X = d_X \leq 2(\Delta - 1) - 2 + 1.$$

Lemma 3d) gives an upper bound on direct debits of at most $d_X \leq 2(\Delta - 1) - 2$. Thus, if there is no outgoing indirect transfer, then the claimed bound on $d_X - c_X$ follows because credits c_X are always nonnegative.

Assume otherwise and let u be the node with an outgoing indirect transfer. Then u has no direct debits by (c5). Moreover, v has by the degree constraint at most $\Delta - 1$ direct debits, call them $vw_1, \dots, vw_{\Delta-1}$. Recall from (c6) that v cannot have outgoing indirect transfers.

We show in Lemma 7b) that at most $\Delta - 2$ of the w_i belong to $\frac{1}{2}$ -paths which need an incoming indirect transfer, hence at most $\Delta - 2$ indirect transfers leave u . So $d_X \leq (\Delta - 1) + (\Delta - 2) = 2(\Delta - 1) - 1$ holds, which gives us the claimed bound on the balance. We prepare the bound of Lemma 7.

Lemma 6. *Let (u, v) be an M -edge and v have direct debits vw_1, \dots, vw_n . Assume that indirect transfers uw_1, \dots, uw_n are added. Before u, v are matched, the degrees of all w_i drop to $d(w_i) = 1$ in the same step s_\downarrow .*

Proof. By definition of indirect transfers, all recipients w_1, \dots, w_n are isolated when u and v are matched with each other. We denote this step by s . By (c3) each node w_i has degree one at the beginning of step s . By (c4), for all i the degree of w_i dropped to $d(w_i) = 1$ in a step s_\downarrow^i with $s_\downarrow^i < s$, when an augmenting path was created by picking an M -edge e_i . Recall that step s_\downarrow^i selected a node of degree at least two.

Assume that there is a step when we have $d(w_j) = 1$ and $d(w_k) \geq 2$ for $j \neq k$. Until w_j is isolated in step s , MINGREEDY picks only nodes of degree one and hence step s_\downarrow^k does not happen until after step s . Therefore $d(w_k) \geq 2$ holds when w_j is isolated, a contradiction since all w_i are isolated in step s when all $d(w_i) = 1$. So all $d(w_i)$ are decreased to $d(w_i) = 1$ by the same step $s_\downarrow = s_\downarrow^1 = \dots = s_\downarrow^k$ creating an augmenting path by picking an M -edge $e_1 = \dots = e_k$. \square

Lemma 7. *Let (u, v) be an M -edge and assume that indirect transfers uw_1, \dots, uw_n are added. Denote by s_\downarrow the step when the degrees of all w_i drop to $d(w_i) = 1$. Then:*

- a) *In step s_\downarrow for every w_i an incident F -edge is removed.*
- b) *It holds that $n \leq \Delta - 2$.*
- c) *If $n = \Delta - 2$, then step s_\downarrow removes exactly $\Delta - 2$ edges of F .*

Proof. We prove a). By (c4) step s_\downarrow creates an augmenting path X , since at step s_\downarrow the degrees of the w_i drop to $d(w_i) = 1$. First consider the case that w_i is not an endpoint of the augmenting path X . Then the edge that connects w_i to its neighbor in X is an F -edge. Now let w_i be an endpoint of X . Since in step s_\downarrow the algorithm selects a node of degree at least two, we have $d(w_i) \geq 2$ at that time. But w_i is incident to at most one edge of X , its M^* -edge, and hence to at least one F -edge.

We prove b) and c). W.l.o.g. step s_\downarrow selects u_\downarrow and matches u_\downarrow with v_\downarrow . Step s_\downarrow selects u_\downarrow when $2 \leq d(u_\downarrow) \leq 3$ since an augmenting path is created and the degrees of the w_i drop to $d(w_i) = 1$ from at most $d(w_i) \leq 3$.

Assume that $d(u_\downarrow) = 2$. Node u_\downarrow is incident only to its M - and M^* -edge. So the n distinct F -edges being removed by a) are incident to v_\downarrow . Since v_\downarrow is also incident to its M - and M^* -edge, we get $n \leq d(v_\downarrow) - 2 \leq \Delta - 2$. Exactly n many F -edges, namely $(v_\downarrow, w_1), \dots, (v_\downarrow, w_n)$, are removed, which holds in particular for $n = \Delta - 2$.

Assume that $d(u_\downarrow) = 3$. Let X_i be the $\frac{1}{2}$ -path of w_i . Step s_\downarrow does not pick the M -edge of one of the X_i , since otherwise an M^* -endpoint w' of the created $\frac{1}{2}$ -path would have to get isolated

by (c1), implying the contradiction $d(w') \leq 2 < d(u_\downarrow)$. So $X \neq X_i$ for all i . At step s_\downarrow we have $d(w_i) \geq d(u_\downarrow) = 3$ for all i . Each w_i is connected to each of $u_\downarrow, v_\downarrow$, since otherwise the degree of w_i could not drop to $d(w_i) = 1$. Hence at step s_\downarrow each of $u_\downarrow, v_\downarrow$ is incident to n edges of F . Again we get $n \leq \Delta - 2$, since each of $u_\downarrow, v_\downarrow$ is also incident to its M -edge and M^* -edge. Assume that $n = \Delta - 2$ holds. Using $\Delta \geq 4$ we get $d(u_\downarrow) = 2 + \Delta - 2 > 3 = d(u_\downarrow)$, a contradiction. (Hence if $n = \Delta - 2$, then case $d(u_\downarrow) = 2$ applies, where exactly n many F -edges are removed.) \square

2.3.4 The Upper Bound on the Balance of Augmenting Paths

For an augmenting path X with $m_X \geq 2$ we claim in Eq. (5) a balance of at most

$$d_X - c_X \leq 2 \cdot (m_X - 1) \cdot (\Delta - 2) - 1.$$

Recall that the nodes of the M -edge that was picked upon creation of X have no outgoing transfers at all (cp. (c7)). For the other $m_X - 1$ edges both nodes are incident to at most $\Delta - 2$ edges in F each, and these edges could move M -funds out of X . In particular, we have already argued that indirect transfers do not increase the overall number of transfers out of an M -edge: If $(u, v) \in M$ and u has outgoing indirect transfers, then by (c5) and (c6) node u has no direct debits and v has no outgoing indirect transfers. In particular, the number of indirect transfers leaving u is bounded above the number of direct debits to v , which in turn is at most $\Delta - 2$.

If one of the M -covered nodes that was not matched upon creation of X has less than $\Delta - 2$ outgoing transfers, then the upper bound on the balance claimed in Eq. (5) is implied. Hence we assume from now on that X has $2 \cdot (m_X - 1) \cdot (\Delta - 2)$ outgoing transfers. Then the claim follows from the next lemma.

Lemma 8. *Let X be an augmenting path with $m_X \geq 2$. If X has its maximum number of $d_X = 2 \cdot (m_X - 1) \cdot (\Delta - 2)$ outgoing transfers, then X has at least one direct credit.*

Proof. Let w denote an endpoint of X that is not isolated upon creation. Lemma 5 states that such a node w exists. Moreover, Lemma 5 also states that X receives the desired M -fund via a direct transfer to w , if w is not incident to its M^* -neighbor w^* in the step s when w is isolated. Thus, we assume in the sequel that w becomes isolated when the M -edge (x, w^*) is picked. Note that (x, w^*) is not the first edge picked in X according to the choice of w . If w is adjacent to x , then the F -edge (x, w) provides the desired direct transfer from x to w .

But if w is not adjacent to x , then w has degree one in step s . Hence x must also have degree one, because w^* is still adjacent to w and x , and thus x is a node of minimum degree. Then x has no direct debits, but it might have indirect transfers to path endpoints that are adjacent to w^* . Since we assumed that X has its maximum number of outgoing transfers, node x has exactly $\Delta - 2$ outgoing indirect transfers; call them $xw_1, \dots, xw_{\Delta-2}$. If there is a w_k that belongs to X , then w_k receives a direct transfer from w^* , which provides the desired credit to X . Thus, we assume from now on that all w_i do not belong to X . It is crucial to note that $w \neq w_i$ for all i because w is in X .

By Lemma 6, the degrees of all w_i drop to $d(w_i) = 1$ in a step $s_\downarrow < s$, and by (c4) step s_\downarrow creates an augmenting path. So at step s we have $d(w) = d(w_1) = \dots = d(w_{\Delta-2}) = 1$. Since the degree of w in the input G is at least two by Lemma 1, there is a step s'_\downarrow with $s'_\downarrow < s$ when the degree of w is decreased to $d(w) = 1$ by removing an incident F -edge.

- Assume that $s'_\downarrow = s_\downarrow$. Since step s'_\downarrow does not remove the M^* -edge (w, w^*) , which is removed later in step s , step s'_\downarrow removes an edge of F incident to w . Also, by Lemma 7a), step s'_\downarrow

removes $\Delta - 2$ edges of F incident to the endpoints w_i . But since $w \neq w_i$, for all i , step s'_\downarrow removes at least $\Delta - 1$ many edges of F . A contradiction to Lemma 7c).

- Assume that $s'_\downarrow \neq s_\downarrow$. Since in step s'_\downarrow the degree of w drops to $d(w) = 1$ and step s_\downarrow selects a node of degree at least 2, step s_\downarrow happens before step s'_\downarrow . Since step s_\downarrow decreases the degrees of all w_i to $d(w_i) = 1$, step s'_\downarrow is a degree-1 step, i.e. a step when a node of degree one is picked. Consequently, step s'_\downarrow does not create a new augmenting path, especially not X . Also, we claim that step s'_\downarrow does not pick any other M -edge of X : To see this, recall that step s'_\downarrow does not remove (w, w^*) , which is removed by step s . Nor does s'_\downarrow remove an edge of F incident to w , since such an edge would imply an ‘internal’ direct transfer.

Hence step s'_\downarrow picks the M -edge of a component other than X . We already argued that s'_\downarrow does not create an augmenting path, thus there is a direct credit to w coming from the M -edge picked by step s'_\downarrow .

□

3 Inapproximability Bounds for Priority Algorithms

In order to study the limitations of greedy algorithms, we utilize the model of priority algorithms. The crucial idea is to regard the input graph \mathcal{I} as a collection of data items, where in the *vertex model* a data item corresponds to a node u in \mathcal{I} with its respective neighbors v_1, \dots, v_d . The data item is denoted by $\langle u; v_1, \dots, v_d \rangle$. An *adaptive priority algorithm* chooses an ordering π on the set of all possible data items, i.e. without actually looking at \mathcal{I} . Then it receives the first data item d of \mathcal{I} w.r.t. π such that the node u of d is still matchable, i.e. neither already matched nor isolated. Now the algorithm has to make an irrevocable decision: Either it chooses a matchable neighbor v and matches u to v , or u becomes isolated and cannot be matched afterwards. Then the algorithm iterates until no matchable nodes are left. Such an algorithm is called *greedy* if it may not choose to isolate u . All orderings and decisions are performed *deterministically*, but may take into account all information about \mathcal{I} gathered so far. In particular, priority algorithms are not resource-bounded.

The adaptive priority game is a convenient way to present inapproximability results by turning the above definition into a game between the algorithm A and an adversary B (see [6] for a primer). Initially, B selects a private graph \mathcal{I} , then the game proceeds in rounds until no matchable nodes are left in \mathcal{I} : In each round A submits an ordering π on all possible data items and receives the first data item of a matchable node u from B . Then A makes an irrevocable decision for u , thereby ending the round.

Angelopoulos and Borodin [1] introduced *fully randomized priority algorithms*: These algorithms proceed like adaptive priority algorithms, but may utilize randomness when determining an ordering of the data items and making decisions.

The class of fully randomized algorithms is quite comprehensive, as it contains for instance the algorithms GREEDY, MINGREEDY, MRG, RANKING, and the KARPSIPSER algorithm [31]. An exception is MDS that we studied in Theorem 2.

First we study deterministic priority algorithms and show an inapproximability bound for adaptive priority algorithms. The underlying construction will provide the basis for our investigation of fully randomized priority algorithms.

Theorem 5. *No adaptive priority algorithm, whether greedy or not, achieves approximation ratio better than $\frac{2}{3}$ in the vertex model.*

The bound holds for graphs with maximum degree three, and hence the deterministic MIN-GREEDY is an optimal adaptive priority algorithm for these graphs.

Proof. Given such a deterministic algorithm A , we consider the two input graphs in Fig. 3 and Fig. 4, both with perfect matching. When the game starts, A submits an ordering π_1 on the set of all data items: Depending on π_1 , the first data item d gives a node of degree two or three; let d be $\langle u; v, w \rangle$ or $\langle u; v, w, z \rangle$. In both cases, if A decides *not* match u , then A will not obtain a matching

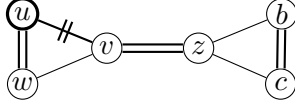


Figure 3: Algorithm A receives $\langle u; v, w \rangle$

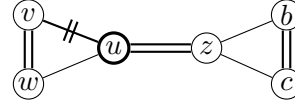


Figure 4: Algorithm A receives $\langle u; v, w, z \rangle$

larger than two edges, and approximation ratio at most $\frac{2}{3}$ follows.

How does the adversary proceed if A matches u ? A has no knowledge about the graphs and we may assume that u matches v . In particular, both graphs are indistinguishable for A . If u has degree two, then the input is the graph in Fig. 3. Otherwise it is the one in Fig. 4. Thus, after matching (u, v) , A can match only one more edge, therefore the claimed bound follows. The inapproximability bound matches the guarantee for the deterministic variant of MINGREEDY given in Theorem 3 for graphs of maximum degree three. \square

Theorem 6. *No fully randomized priority algorithm can achieve an expected approximation ratio better than $\frac{5}{6}$ for the vertex model.*

Proof. We apply Yao's Minimax Principle [44]. We have to construct a hard distribution over input instances and to analyze the best deterministic algorithm (that knows the distribution). As distribution we take all the graphs corresponding to the permutations of the node labels of Fig. 3. We will consider only mistakes made in the first round and assume that the algorithm proceeds optimally afterwards.

First note that if the algorithm decides to isolate the node given in the first round, it cannot obtain a matching larger than two. Thus, we may assume that the first node is matched. Furthermore, if the first matching is non-optimal, the algorithm again obtains at most two edges and has approximation ratio at most $\frac{2}{3}$. On the other hand, if the first matching is optimal, a maximum matching can be obtained.

Since we are in the first round, the algorithm has no information which neighbor is the optimal choice, and any neighbor is the optimal mate with same probability because we picked a labeling of the nodes uniformly at random. Thus, the best strategy for the algorithm is to request no degree three prior to degree two, since the probability of matching the first node optimally decreases with its degree, and the bound follows because a node of degree two is matched optimally with probability $\frac{1}{2}$. The bound is still valid if the number of nodes, the number of edges, and all degrees are revealed in advance by the adversary. \square

We compare our inapproximability result for fully randomized algorithms to the bounds obtained by Goel and Tripathi [22]. On the one hand, they studied randomized greedy algorithms in

the *oblivious query commit model*. In this model edges are not revealed to the algorithm; the only way to figure out whether a particular edge exists is to probe the pair of its endpoints. If an edge is found whose endpoints are both not matched yet, it must be added to the matching irrevocably. In this case both nodes are removed from the graph.

Note that it is impossible to design an algorithm that chooses nodes depending on their degrees, and in particular the algorithm cannot select a node of degree one. That's why none of these randomized algorithms achieves an expected approximation ratio better than $\frac{19}{24} \approx 0.792$ on a triangle with a single edge attached.

On the other hand, Goel and Tripathi consider the more restricted class of *vertex iterative* algorithms. A vertex iterative algorithm picks randomly a vertex, say u , in each round and then may scan (a subset of) the other vertices, one after another, to check whether they are adjacent to u . As required by the oblivious query commit model, whenever an edge is found, it is added to the matching. Moreover, before every probe the algorithm may choose to isolate the currently inspected vertex u irrevocably and thereby skip to the next round.

Goel and Tripathi show that no vertex iterative algorithm obtains an expected approximation ratio better than $\frac{3}{4}$ on the graph of Dyer and Frieze [15].

The MRG algorithm and RANKING are prominent representatives of vertex iterative algorithms. MINGREEDY, however, cannot be implemented in the oblivious query commit model.

We point out that the class of fully randomized adaptive priority algorithms in the vertex model contains all randomized algorithms in the oblivious query commit model.

Theorem 7. *Every algorithm in the oblivious query commit model (and hence every vertex iterative algorithm) can be implemented as fully randomized priority algorithm in the vertex model.*

On the other hand, the fully randomized priority algorithm MINGREEDY cannot be implemented in the oblivious query commit model.

Proof. We have already pointed out that the oblivious query commit model does not allow the algorithm to select edges depending on the degrees of their nodes. In fact, the lower bound given in [22] relies on this observation. In the sequel we show the first claim of the lemma.

Recall from the definition of vertex iterative algorithms that each such randomized algorithm can be implemented in the oblivious query commit model. Thus, it suffices to demonstrate that an algorithm A in the oblivious query commit model can be simulated by some fully randomized greedy algorithm B . Let G be the input on which both algorithms are run. Since G is unknown to both algorithms, we assume for the sake of convenience that both algorithms are provided a set of possible node identifiers in advance. In particular, we assume that an upper bound on the number of nodes is common knowledge.

We assume that all random bits that A uses are drawn in advance. Thus, given these random bits B can simulate A on G deterministically. Algorithm B determines the first query of A , say for (u, v) . B simulates the first query by giving highest priority to all possible data items of node u that contain v as neighbor. Since exactly one of these data items is consistent with G , their ordering w.r.t. each other is irrelevant. Let π_1 be the ordering submitted by B in the first round. If (u, v) exists, A adds this edge to its matching, and so does B .

If (u, v) does not exist, then A may query probabilistically for another edge, say (x, y) . But B will receive the first data item according to π_1 that exists in the graph. In particular, the priority algorithm may only change its ordering of data items after it has received a data item; since there is no such data item for u in G , B may take no action at this time. Fortunately, B has access to the

description of A and its pool of random bits, hence B can determine a priori which edge A would query next if (u, v) does not exist. Therefore, in the first round B plans ahead and enumerates all data items for node x that contain y as neighbor after the prefix of the data items for u in π_1 . We iterate this process.

Assume that A eventually finds some edge (u', v') and adds it to the matching. Then B receives the data item for u' and picks the same edge. The gist is that B can infer the same information from the data item for u' that A has gathered: The position of the data item in π_1 implies that no edge of higher priority exists in G . Moreover, the data item of u' contains at least the information that (u', v') exists, and perhaps additional information about the neighborhood of u' .

Thus, B has always at least the same knowledge about G and hence can simulate A subsequently. Here it is crucial to note that according to the respective definitions of their models, A and B may only query data items of nodes that have not been matched yet.

Once B has decided the first data item, it determines π_2, π_3 and so on analogously. \square

3.1 Greedy Adaptive Priority Algorithms and Degree Bounded Graphs

The inapproximability bounds given in Sect. 3 rely on graphs with maximum degree three. How do priority algorithms perform when applied to arbitrary graphs?

Recall that *greedy* adaptive priority algorithms do not have the option to isolate the node given in the current data item; they must add an edge to their matching in each round. We show that every such algorithm has approximation ratio at most $\frac{1}{2} + \varepsilon$ for any $\varepsilon > 0$. Thus, randomness seems essential for greedy algorithms in order to achieve a non-trivial guarantee.

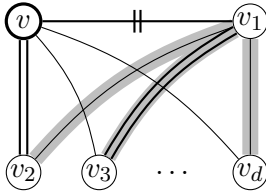


Figure 5: A connected component of G : Gray edges are unknown to algorithm A

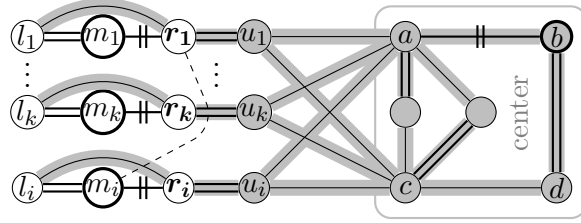


Figure 6: The full construction: Gray nodes and edges are unknown to A , frontier nodes are drawn bold, the dashed edge is an example for edge $(m_i = v, r_j)$ in a type 3 round

Theorem 8. *Let A be a greedy adaptive priority algorithm. There is a graph with maximum degree at most Δ , for which the approximation ratio of A is not better than $\frac{\Delta-1}{2\Delta-3}$.*

Proof. The adversary creates the input graph on-the-fly during the adaptive priority game; this is legal as long as the adversary ensures that the final graph is consistent with the revealed data items, since the algorithm works deterministically. We call a node *known*, if it was contained in a data item shown in a previous round, and *unknown* otherwise. The gist of the construction is that all known nodes are either already matched or isolated.

The game between A and the adversary B consists of two phases: The *regular game* lasts for $s = \Delta - 3$ rounds, the *endgame* has two rounds. We consider round $i \geq 1$ of the regular game and assume that B returns data item a_i that belongs to one of the following three types; B will give no other data item to A .

Type 1: $a_i = \langle v; v_1, \dots, v_d \rangle$ with $3 \leq d \leq \Delta$ and all nodes are unknown. Thus, the nodes are indistinguishable and we may assume that A matches v to v_1 . B constructs a separate connected component C (cf. Fig. 5): The optimum M^* contains two edges $(v, v_2), (v_1, v_3)$ in C , whereas A adds only the edge (v, v_1) to its matching M . Observe that all (data items of) nodes in C belong to type 1 or 2 before the current round and are either matched or isolated afterwards.

Type 2: $a_i = \langle v; v_1, v_2 \rangle$, and all nodes are unknown. Assume that A matches v with v_1 . B constructs a triangle $\{l_i, m_i, r_i\}$ with $l_i=v_2, m_i=v, r_i=v_1$ and an edge (r_i, u_i) with a new node u_i , which connects the triangle to the unknown center (cf. Fig. 6). The center will connect triangles created for data items of types 2 and 3. Again M^* is extended by two edges $(l_i, m_i), (r_i, u_i)$, whereas only (m_i, r_i) is added to M . To verify the legality, we observe that before m_i, r_i are matched, (the data items of) 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.

Type 3: $a_i = \langle v; v_1, v_2, v_3 \rangle$, where v, v_1, v_2 are unknown and v_3 is known. Node v_3 occurred in a data item presented previously, and in particular must be a neighbor of some node r_j (with $j < i$) by construction. But then, does $v = u_j$ hold? Not necessarily, since B may introduce further neighbors of r_j , since r_j was matched on its first appearance and hence its data item is never presented to the algorithm. Since $v_3 = r_j$ is already matched and v_1, v_2 are both unknown, we may assume that A matches v to v_1 . Again B creates a triangle $\{l_i = v_2, m_i = v, r_i = v_1\}$ and an additional edge (r_i, u_i) with a new node u_i . Moreover, B also inserts the edge $(m_i=v, r_j)$ to preserve consistency. M^* (resp., M) is extended by $(l_i, m_i), (r_i, u_i)$ (resp., (m_i, r_i)). Before m_i, r_i are matched, node l_i is of type 2, nodes r_i, u_i are of type 1 and $m_i=v$ is of type 3. After matching m_i, r_i , nodes l_i, m_i, r_i are isolated and u_i turns into type 3. u_j is still of type 3.

The regular game ends after round $s = \Delta - 3$. We consider the graph created by B (cf. Fig. 6): a, b are of type 1 (for a specific value of d), b, c of type 2, and the u -nodes of type 3. The edges matched in the endgame are all incident in the center: B enforces that A matches only two edges, whereas the optimum obtains three. Summing up, we have $|M| = s + 2 = \Delta - 1$ and $|M^*| = 2s + 3 = 2\Delta - 3$, and the claim follows. B asserts that the edge (a, b) is matched in round $\Delta - 2$; afterwards c can be matched to some neighbor, leaving all other nodes isolated. We distinguish the following types for data item $a_{\Delta-2}$:

I) $a_{\Delta-2} = \langle v; v_1, \dots, v_d \rangle$ is of type 1. Since no nodes in $a_{\Delta-2}$ is known, we assume that A matches v to v_1 . B chooses $v = a, v_1 = b$, and v_2, \dots, v_d as the remaining neighbors of a .

II) $a_{\Delta-2} = \langle v; v_1, v_2 \rangle$ is of type 2. Again we may assume that (v, v_1) is matched, hence B chooses $v = b, v_1 = a$, and $v_2 = d$.

III) $a_{\Delta-2} = \langle v; v_1, v_2, v_3 \rangle$ is of type 3. As above, the known node v_3 is some matched node r_j , $j < \Delta - 2$, and we may assume that (v, v_1) is matched by A . The adversary chooses $v_3 = r_j, v=b, v_1=a$, and $v_2=d$; therefore, B creates the edge (v_3, b) (not present in Fig. 6).

Concludingly, we verify that no node has degree larger than Δ : Nodes in type 1 components have degree at most Δ by definition of the component. The degree of a and c is at most $\Delta = 3 + s$, since each round of the regular game adds at most one u -neighbor to both. At most $s - 1$ neighbors are added to an r -node during the regular game, at most one neighbor is added in (the first step of) the endgame, hence degrees of r -nodes are at most Δ as well. All other nodes have degree at most three. \square

4 Hypergraph Matching

We study the limitations of greedy algorithms for the more general k -Hypergraph Matching Problem. In a k -hypergraph an edge may have up to k nodes. The goal is to find a maximum set of node disjoint edges. As for common graphs, a $\frac{1}{k}$ -approximation is easily obtained by greedily picking edges [33]. We show that greedy adaptive priority algorithms in the vertex model cannot surpass this trivial worst case guarantee.

k -hypergraph matching is NP-complete: 3-dimensional matching, where each edge has exactly three nodes and the graph is tripartite, as well as the unrestricted hypergraph matching problem, also called the set packing problem, belong to Karp's 21 NP-complete problems. For an overview of problems closely related to hypergraph matching, see Chan and Lau [10].

We consider k -uniform hypergraphs where each edge has exactly k nodes. To achieve non-trivial approximation guarantees efficiently, local search was shown to be successful. Hurkens and Schrijver [29] 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 [12] recently improved the approximation ratio to $\frac{k+1+\varepsilon}{3}$. On the other hand, Hazan, Safra, and Schwartz [25] showed that k -uniform hypergraph matching cannot efficiently be approximated within a factor of $O(\frac{k}{\ln k})$.

Greedy approaches have also been investigated. Bennett and Bohman [4] showed the following bound on the expected performance of GREEDY on k -uniform D -regular hypergraphs H 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 $(L/D)^{\frac{1}{2(k-1)} + o(1)}$ of the nodes remains unmatched whp. Aronson et al. [2] investigated GREEDY on general 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. We give a tight bound for greedy adaptive priority algorithms.

Theorem 9. *No greedy adaptive priority algorithm in the vertex model has approximation ratio better than $\frac{1}{k}$ for k -uniform hypergraph matching with $k \geq 3$.*

Proof. Given an algorithm A , we construct a k -uniform hypergraph on which A has approximation ratio exactly $\frac{1}{k}$. The instance constructed by the adversary is illustrated in Fig. 7. The white (vertical) edges constitute a maximum matching $\{e_0, \dots, e_{k-1}\}$. The topmost horizontal edge e will be the only edge picked by the adaptive priority algorithm. (We call a node an e -node if it belongs to the edge labeled e , and a non - e -node otherwise.)

The adversary creates $k - 1$ additional edges that are depicted as gray edges in Fig. 7. For $0 \leq i \leq k - 2$ an edge contains the (unique) e -node of vertical edge e_i and a non - e -node of each $e_j \neq e_i$ chosen in a way such that all non - e -nodes are covered at most once. Exactly one non - e -node of each of e_0, \dots, e_{k-2} is not contained in a gray edge, call them v_0, \dots, v_{k-2} .

So far, the e -nodes of e_0, \dots, e_{k-2} have degree three, the e -node of e_{k-1} has degree two. The adversary creates $k - 1$ more edges, that are displayed as black (vertical) lines in Fig. 7. These edges use $K = \frac{(k-1)(k-2)}{2}$ new nodes $1, 2, \dots, K$. Using these nodes, the adversary creates sets S_0, \dots, S_{k-2} of $k - 2$ nodes each, such that every new node occurs in exactly two of the S_i and $|S_i \cap S_j| = 1$ whenever $i \neq j$. Refer to Fig. 8 for the construction of the S_0, \dots, S_{k-2} : a node listed in the j -th row has its second occurrence in the j -th column. The i -th new edge, with $0 \leq i \leq k - 2$, contains v_i , the e -node of $e_{i+1 \bmod k-1}$, and the nodes of S_i .

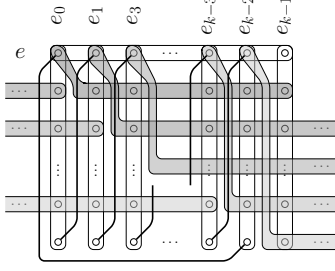


Figure 7: A hard k -uniform hyper-graph matching instance

$$\begin{aligned}
 S_0 &= \{ 1, 2, 3, 4, \dots, k-2 \} \\
 S_1 &= \{ 1, k-1, k, k+1, \dots, 2k-5 \} \\
 S_2 &= \{ 2, k-1, 2k-4, 2k-3, \dots, 3k-9 \} \\
 S_3 &= \{ 3, k, 2k-4, \dots \} \\
 &\vdots \\
 S_{k-2} &= \{ k-2, 2k-5, 3k-9, \dots, K \}
 \end{aligned}$$

Figure 8: Definition of the S_i

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 the new nodes in S_0, \dots, S_{k-2} , have degree two.
- iii. Any two edges have at most one node in common.
- iv. The edge e shares exactly one node with any other edge.

Now the data items of the input graph look as follows: The data item

$$\langle u; V_1, \dots, V_d \rangle$$

of node u lists the d hyperedges incident in u : each hyperedge $\{u\} \cup V_i$ is represented by the node set V_i .

How does the greedy adaptive priority algorithm A proceed when the adaptive priority game starts? Recall that A submits an ordering π on the set of all data items *without looking at the graph*. In the first round the adversary presents the, according to π , first data item $\langle u; V_1, \dots, V_d \rangle$ with $d \in \{2, 4\}$ (which are the only degrees present in the graph, by i. and ii.), $V_i \cap V_j = \emptyset$ for $i \neq j$ (node u is the only common node of all incident edges, by iii.) and $|V_i| = k-1$ for all i (the graph is k -uniform). Since A is greedy, A selects an incident edge $\{u\} \cup V_i$ and adds it to its matching.

First assume that $d = 4$ holds. Then the adversary may relabel the nodes in the instance such that u is the e -node of e_0 , since this is the first data item revealed to the algorithm. The greedy adaptive priority algorithm must pick an edge incident to u , and the adversary asserts that this edge is e . The matching is maximal by iv. In case $d = 2$ the adversary relabels the 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 . \square

5 Conclusion

Our inapproximability result for fully randomized priority algorithms implies that greedy-like algorithms cannot compete with algorithms based on augmenting-paths or algebraic methods. Nonetheless, conceptually simple algorithms, that are easy to implement and very efficient in practice, deserve further investigation.

Theorem 8 gives inapproximability bounds for a large class of deterministic greedy algorithms on graphs with maximum degree $\Delta \geq 3$. We conjecture that the deterministic variant of MINGREEDY achieves these bounds for all Δ .

Moreover, our approximation guarantee given in Theorem 3 does not take into account that choosing a random neighbor has a good probability of picking an optimal neighbor, if the degrees are small (cp. [17] and also Sect. 1). We leave it as future work to exploit this observation.

References

- [1] S. Angelopoulos and A. Borodin. Randomized priority algorithms. *Theor. Comput. Sci.*, 411(26-28):2542–2558, 2010.
- [2] J. Aronson, M. E. Dyer, A. M. Frieze, and S. Suen. Randomized greedy matching II. *Random Struct. Algorithms*, 6(1):55–74, 1995.
- [3] J. Aronson, A. M. Frieze, and B. Pittel. Maximum matchings in sparse random graphs: Karp-Sipser revisited. *Random Struct. Algorithms*, 12(2):111–177, 1998.
- [4] P. Bennett and T. Bohman. A natural barrier in random greedy hypergraph matching. *CoRR*, abs/1210.3581, 2012.
- [5] B. Berger, R. Singht, and J. Xu. Graph algorithms for biological systems analysis. In *SODA*, pages 142–151, 2008.
- [6] A. Borodin, J. Boyar, K. S. Larsen, and N. Mirmohammadi. Priority algorithms for graph optimization problems. *Theor. Comput. Sci.*, 411(1):239–258, 2010.
- [7] A. Borodin, I. Ivan, Y. Ye, and B. Zimny. On sum coloring and sum multi-coloring for restricted families of graphs. *Theor. Comput. Sci.*, 418:1–13, 2012.
- [8] A. Borodin, M. N. Nielsen, and C. Rackoff. (Incremental) priority algorithms. *Algorithmica*, 37(4):295–326, 2003.
- [9] T.-H. H. Chan, F. Chen, X. Wu, and Z. Zhao. Ranking on arbitrary graphs: Rematch via continuous lp with monotone and boundary condition constraints. In *SODA*, pages 1112–1122, 2014.
- [10] Y. Chan and L. Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical Programming*, 135(1-2):123–148, 2012.
- [11] Y.-Q. Cheng, V. Wu, R. T. Collins, A. R. Hanson, and E. M. Riseman. Maximum-weight bipartite matching technique and its application in image feature matching. In *In Proc. SPIE Visual Comm. and Image Processing*, 1996.
- [12] M. Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *FOCS*, pages 509–518, 2013.
- [13] S. Davis and R. Impagliazzo. Models of greedy algorithms for graph problems. *Algorithmica*, 54(3):269–317, 2009.

- [14] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.
- [15] M. E. Dyer and A. M. Frieze. Randomized greedy matching. *Random Struct. Algorithms*, 2(1):29–46, 1991.
- [16] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.
- [17] J. Edmonds and D. R. Fulkerson. Transversals and matroid partition. *J. Res. Natl. Bur. Stand.*, 1965.
- [18] A. M. Frieze, A. J. Radcliffe, and S. Suen. Analysis of a simple greedy matching algorithm on random cubic graphs. *Combinatorics, Probability & Computing*, 4:47–66, 1995.
- [19] H. N. Gabow. An efficient implementation of Edmonds’ algorithm for maximum matching on graphs. *J. ACM*, 23(2):221–234, 1976.
- [20] H. N. Gabow. Set-merging for the Matching Algorithm of Micali and Vazirani. *CoRR*, abs/1501.00212v1, 2014.
- [21] J. F. Geelen. An algebraic matching algorithm. *Combinatorica*, 20(1):61–70, 2000.
- [22] G. Goel and P. Tripathi. Matching with our eyes closed. In *FOCS*, pages 718–727, 2012.
- [23] A. V. Goldberg and A. V. Karzanov. Maximum skew-symmetric flows and matchings. *Math. Program.*, 100(3):537–568, 2004.
- [24] N. J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39(2):679–702, 2009.
- [25] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k -set packing. *Computational Complexity*, 15(1):20–39, 2006.
- [26] M. Hosaagrahara and H. Sethu. Degree-sequenced matching algorithms for input-queued switches. *Telecommunication Systems*, 34(1-2):37–49, 2007.
- [27] S. Hougardy. Linear Time Approximation Algorithms for Degree Constrained Subgraph Problems. In W. Cook, L. Lovász, and J. Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 185–200. 2009.
- [28] N. Huang and A. Borodin. Bounds on double-sided myopic algorithms for unconstrained non-monotone submodular maximization. In *ISAAC*, pages 528–539, 2014.
- [29] 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.
- [30] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.
- [31] R. M. Karp and M. Sipser. Maximum matchings in sparse random graphs. In *FOCS*, pages 364–375, 1981.

- [32] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.
- [33] B. Korte and D. Hausmann. An analysis of the greedy algorithm for independence systems. *Annals of Discrete Mathematics*, 2:65–74, 1978.
- [34] J. Magun. Greedy matching algorithms: An experimental study. *ACM Journal of Experimental Algorithmics*, 3:6, 1998.
- [35] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *STOC*, pages 597–606, 2011.
- [36] Z. Miller and D. Pritikin. On randomized greedy matchings. *Random Struct. Algorithms*, 10(3):353–383, 1997.
- [37] M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *FOCS*, pages 248–255, 2004.
- [38] M. Poloczek. Bounds on greedy algorithms for MAX SAT. In *ESA*, pages 37–48, 2011.
- [39] M. Poloczek and M. Szegedy. Randomized greedy algorithms for the maximum matching problem with new analysis. In *FOCS*, pages 708–717, 2012.
- [40] A. E. Roth, T. Sönmez, and M. U. Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151 – 188, 2005.
- [41] G. Tinhofer. A probabilistic analysis of some greedy cardinality matching algorithms. *Annals of Operations Research*, 1:239–254, 1984.
- [42] P. Tripathi. *Allocation problems with partial information*. PhD thesis, Georgia Institute of Technology, 2012.
- [43] V. V. Vazirani. An improved definition of blossoms and a simpler proof of the MV matching algorithm. *CoRR*, abs/1210.4594, 2013.
- [44] A. C.-C. Yao. Lower bounds by probabilistic arguments. In *FOCS*, pages 420–428. IEEE, 1983.

A A Linear Time Implementation of MinGreedy

For MRG and RANKING Poloczek and Szegedy [39] propose a data structure that allows to run both algorithms in linear time. GREEDY can also be implemented in linear time using a data structure similar to the one we describe below.

Given an adjacency list representation of the input graph $G = (V = \{0, \dots, n-1\}, E)$, the data structure can be initialized in linear time $O(|E| + |V|)$. 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.

How is a minimum degree node u 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 ($0 \leq i \leq k$) such that S_i precedes all S_j with $i < j$. Each S_i contains all nodes that currently have degree d_i in contiguous cells of S . A doubly linked list D stores (from head to tail) the borders of S_0, S_1, \dots, S_k . 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 .

To select a random neighbor v of u in constant time, instead of adjacency lists we utilize adjacency *arrays* (which have same lengths as the lists and can be computed in linear time during the preprocessing phase). To pick v from the adjacency array A_u of u , we assert that the currently available neighbors of u are stored in a consecutive part of A_u .

Now that nodes u and v are selected, the edge (u, v) , and all incident edges of u and v are removed from the data structure. We remove these edges one-by-one, each in constant time.

This is how we update the adjacency arrays. Let a node x and an index of a cell in A_x be given, say containing neighbor y . Assume that the array cell in A_x containing node y also stores the index of the array cell in A_y containing node x as a *reference* and vice versa. 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 handle A_y and x analogously. We also update the references of the two moved entries accordingly; this is done in constant time using the references stored inside the moved entries. The references are initialized during the construction of the adjacency arrays.

How to update S and D for the removal of an edge (x, y) ? Since the degrees d_x, d_y of x respectively 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, P_S : $P_D[x]$ stores a pointer to the D -entry containing x , $P_S[x]$ 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} , i.e., the node stored at the smallest index, and x is appended to the “right” of S_{d_x-1} . 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., 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.

Note that S, D, P_S, P_D can be initialized in time $O(|E| + |V|)$ during the preprocessing phase by scanning through the adjacency lists of G a constant number of times.