

Max-Min Greedy Matching

by

Ramazan Rakhmatullin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2024

© Ramazan Rakhmatullin 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

All research detailed in this thesis was completed in collaboration with my supervisor, Kanstantsin Pashkovich.

Abstract

We study the problem of max-min greedy matching introduced in [7]. The max-min greedy matching problem is a variant of the well-known online bipartite matching problem introduced in [13]. Given a bipartite graph $G = (U \cup V; E)$, the first player selects a priority ordering π of V . After that the second player selects an arrival order σ of U depending on the ordering π selected by the first player. Once both players did their selections, the vertices of U arrive in the order specified by σ . Upon arrival, each vertex in U is matched with its highest-priority neighbor that is still unmatched, if one exists. Here, the highest-priority is defined with respect to π . The first player aims to maximize the size of the resulting matching, while the second player seeks to minimize it.

We investigate the max-min greedy matching problem. We aim to improve the lower bound of the guaranteed matching size achievable by the first player. We present a linear time algorithm that constructs an ordering π such that for every σ the ratio between the resulting matching size and the maximum matching size is at least $\frac{5}{9}$. Our result improves the previous best known ratio of $\frac{1}{2} + \frac{1}{86}$ from [7]. The best known upper bound on the ratio is $\frac{2}{3}$ [4]. Thus, our result makes the gap between the lower and upper bounds for the ratio substantially smaller. Our method introduces a novel “balanced path decomposition” technique. This decomposition provides crucial structural properties such as existence of large matchings within these paths; and absence of certain edges between the paths. These structural properties allow us to prove the desired lower bound on the ratio.

Our findings suggest that the achieved matching ratio can be further improved and that the balanced path decomposition may have broader applications.

Acknowledgements

I would like to thank my supervisor professor Kanstantsin Pashkovich and administrative coordinator of graduate studies at C&O department Melissa Cambridge.

Table of Contents

Author's Declaration	ii
Statement of Contributions	iii
Abstract	iv
Acknowledgements	v
1 Introduction	1
1.1 Online Bipartite Matching Problem	1
1.2 Max-Min Greedy Matching	2
1.3 Our Results	4
1.4 Related Work	4
1.4.1 Online Bipartite Matching and Variants	4
1.4.2 Fractional Variant and Hardness for Adversary	5
1.4.3 Resource Allocation and Pricing Mechanisms	6
1.5 Organization	7
2 Preliminaries	9
2.1 Initial Observations and Simplifications	13
2.2 Reduction to Directed Graph	16

3	Main result	20
3.1	Initial Path Decomposition	21
3.2	$\frac{1}{2} - \varepsilon$ Bound	22
3.3	Tightening the Decomposition	25
3.4	$\frac{1}{2} - \varepsilon$ Bound Using a Single Permutation	27
3.5	Special Case of Hamiltonian Graphs	30
3.6	Balanced Path Decomposition	31
3.7	Proving $\frac{4}{9}$ Bound	36
3.8	Further Directions	38
	3.8.1 Imposing an Order Within Sets	38
	3.8.2 Bidirected Graphs	40
4	Conclusion and Further Directions	42
	References	43

Chapter 1

Introduction

1.1 Online Bipartite Matching Problem

In the fast-paced world of online advertising, platforms must swiftly match incoming user impressions with suitable advertisements. Imagine a scenario where users visit a website, and each visit represents an opportunity to display an ad. Advertisers have specific targets and budgets, and the platform must decide in real-time which ad to show to each user. This decision is made without knowledge of future user visits, aiming to maximize overall engagement or revenue. This real-world challenge is an instance of the **online bipartite matching problem**, where one set consists of ads and the other consists of users arriving sequentially.

The online bipartite matching problem was introduced by [13] to model situations where decisions must be made on the fly without complete information about future events. In this problem, one side of the bipartite graph (e.g., ads) is known in advance, while the other side (e.g., users) arrives over time. Upon the arrival of each user, the algorithm must immediately and irrevocably match them to an available ad or decide not to match at all. The objective is to maximize the total number of matches or the total value derived from these matches if edges have associated weights.

This problem is highly relevant in domains requiring real-time decision-making under uncertainty, such as online marketplaces, job scheduling, and network routing. A well-known result in this area is the **Ranking algorithm** [13], which achieves an optimal competitive ratio of $1 - \frac{1}{e}$ (approximately 63.2% of the optimal offline solution). This means that the algorithm's performance is provably close to the best possible, even without

knowledge of future arrivals. The online bipartite matching problem has spurred extensive research, leading to advancements in online algorithms and their applications in various fields where efficient resource allocation is crucial.

1.2 Max-Min Greedy Matching

While the online bipartite matching problem provides a solid framework for modeling sequential decision-making under uncertainty, certain applications require a more strategic approach to ordering resources or tasks to optimize outcomes. This necessity leads us to the **max-min greedy matching problem**, as explored by [7], which introduces an element of adversarial ordering into the matching process.

Consider an example in task assignment within a corporate setting. Suppose there is a set of workers V and a set of jobs U , where each job requires specific skills or qualifications that only certain workers possess. The compatibility between workers and jobs is represented by a bipartite graph $G(U \cup V; E)$, which is known in advance and admits a perfect matching.

The manager assigns monetary values to jobs, ordering them from most favorable (highest pay) to least favorable (lowest pay). However, the workers arrive in an arbitrary (adversarial) order and choose the highest-paying job they are qualified to perform. The challenge is to determine a job ordering that ensures a high number of assignments and job completions, regardless of the sequence in which workers arrive.

In the traditional online bipartite matching problem, **the algorithm does not know the preference graph G in advance**. Vertices from one side (e.g., workers) arrive sequentially, each revealing their set of compatible matches upon arrival. **The arrival order σ is fixed but unknown to the algorithm**. At each step, the algorithm must make immediate and irrevocable matching decisions without knowledge of future arrivals. Both G and σ may be arbitrary and are chosen in advance, but the algorithm has no prior information about them. Due to this inherent uncertainty, [13] demonstrated that any algorithm achieving a meaningful performance **must incorporate randomness** in its decision-making process. Consequently, the goal is to optimize the **expected performance under uncertain input** — maximizing the average number of matches over all random choices made by the algorithm for a fixed G and σ , despite what G and σ actually are.

In contrast, the **max-min greedy matching** problem assumes that the entire **preference graph G is known beforehand**. However, **the arrival order σ** of one side (e.g.,

workers) **is determined adversarially** and may be chosen after the algorithm commits to its strategy. The algorithm can control the order π of the other side (e.g., jobs) and aims to find an ordering that maximizes the number of matches, regardless of how the adversary orders the arrivals. The objective here is to optimize the **worst-case performance** — ensuring that even under the most unfavorable arrival sequence, the number of matches achieved is as high as possible. Even with the benefit of knowing the graph, the shift from optimizing expected outcomes to optimizing the performance against adversarial conditions introduces new challenges and requires different algorithmic strategies.

An immediate observation in the max-min greedy matching problem is that any algorithm can guarantee matching **at least half** of the optimal number of matches — that is, at least half the size of a maximum matching. See Lemmas 2.1.2, 2.1.3, 2.1.4 for more details. A natural question arises: can we surpass this baseline under adversarial conditions?

[7] address this challenge by exploring whether there exists a linear order π over V such that the greedy matching algorithm, when confronted with any adversarial arrival order σ over U , matches **strictly more than half** of the vertices. They affirmatively answer this question by proving that there exists a polynomial time algorithm to compute such an order π . Specifically, they show that it is possible to ensure that at least a fraction $\rho > 0.51$ of the vertices in V are matched, regardless of the minimizing player’s strategy. This result is significant because it demonstrates that, even under the worst-case arrival order, one can guarantee a matching that exceeds half of the maximum possible matches. It improves upon the baseline guarantee of 0.5 and highlights the potential for strategic ordering to enhance performance in adversarial settings.

An important aspect of the max-min greedy matching problem is establishing upper bounds on the performance ratio — that is, determining the maximum fraction of the optimal matching size that can be guaranteed under adversarial conditions. This upper bound represents a limitation: no matter what strategy the maximizing player employs, the minimizing player can always respond in a way that prevents surpassing this ratio. [4] established an upper bound of $\frac{2}{3}$ for this problem by constructing a specific G where, for any ordering π chosen by the maximizing player, the minimizing player can find an arrival order σ that limits the matching size to at most $\frac{2}{3}$ fraction of the offline maximum matching. See Lemma 2.1.5 for the details of constructing G .

The ultimate goal in this line of research is to close the gap between the lower and upper bounds of the achievable performance ratio. This involves two complementary approaches: developing algorithms that guarantee higher matching ratios (thus improving the lower bound) and finding counterexample graphs that demonstrate the impossibility of achieving ratios above a certain threshold (thereby tightening the upper bound).

1.3 Our Results

As the main result of our work, we present an algorithm that runs in linear time relative to the size of the input graph and produces an ordering π that guarantees a matching ratio of $\frac{5}{9}$ (approximately 0.55). This improves the previous best result of $\frac{1}{2} + \frac{1}{86}$ [7] and brings the lower bound closer to the known upper bound of $\frac{2}{3}$ (approximately 0.66) [4]. Our approach builds upon the insights of [7], who highlighted the significance of long paths in the max-min greedy matching problem. We introduce a novel path decomposition technique called the **balanced path decomposition**, which ensures the existence of large matchings corresponding to these paths while simultaneously preventing certain edges between paths. These properties enable us to construct an ordering π that achieves the improved matching ratio. This line of work is detailed in Chapter 3.

We believe that the achieved ratio of $\frac{5}{9}$ can be improved further. We also believe that the balanced path decomposition may be of independent interest. Both the constructed ordering π and the decomposition itself may possess additional properties worth studying.

1.4 Related Work

1.4.1 Online Bipartite Matching and Variants

The online bipartite matching problem [13] has been a cornerstone in the study of online algorithms and competitive analysis. [13] proposed the Ranking algorithm, which assigns random priorities to items and matches each arriving agent to the highest-ranked available item. They show that this algorithm achieves a competitive ratio of $1 - \frac{1}{e}$ and that no algorithm for online bipartite matching has a competitive ratio better than $1 - \frac{1}{e} + o(1)$. Subsequent research has explored various aspects of online bipartite matching, including deterministic algorithms, regular graphs, and stochastic input models [3, 10, 12, 17].

For instance, in the case of d -regular bipartite graphs, [3] developed a randomized algorithm that achieves an expected competitive ratio approaching 1 as the degree d increases. Specifically, their algorithm attains a competitive ratio of $1 - O(\sqrt{\log d/d})$ in expectation, with a lower bound of $1 - O(1/\sqrt{d})$. This contrasts with the max-min greedy matching problem where, even for regular graphs with arbitrarily large degrees, the guaranteed competitive ratio is bounded above by $\frac{8}{9}$ [7].

In the random arrival model, where the order of arriving vertices is uniformly random, the deterministic greedy algorithm achieves a competitive ratio of $1 - \frac{1}{e}$ [10]. No determin-

istic algorithm can exceed a ratio of $\frac{3}{4}$ in this setting [10]. The Ranking algorithm, when adapted to random arrivals, achieves a competitive ratio between 0.696 [16] and 0.727 [12], while the upper bound for any randomized algorithm is 0.823 [17].

The study of online bipartite matching has significant applications in online advertising, task assignment, and resource allocation in networks. For comprehensive surveys on this topic, see [18] and [1].

While max-min greedy matching problem was initially proposed as a variation of online bipartite matching problem, these are two fundamentally different mathematical models. In the max-min greedy matching problem

- The algorithm operates under complete information about the graph G .
- The algorithm is effectively restricted to deterministic strategies.
- The adversary chooses arrival order σ after seeing priority order π constructed by the algorithm;

while in the online bipartite matching problem

- The algorithm has no information about the graph G .
- The algorithm is allowed to use randomized strategies; and its performance is calculated using the expectation over the randomness in such a strategy.
- The adversary chooses arrival order σ without seeing the priority order π constructed by the algorithm.

Therefore neither the lower bounds nor the upper bounds known for the online matching problem can be immediately applied to the max-min greedy matching problem.

1.4.2 Fractional Variant and Hardness for Adversary

Recent research has introduced significant developments to the max-min greedy matching problem by exploring new variants and providing deeper insights into the computational complexities involved.

[2] consider the problem from the adversary’s perspective by defining **the adversarial order minimum matching problem**. In this variant, given a bipartite graph $G(U \cup V; E)$

and a fixed item permutation π chosen by the algorithm, the adversary seeks a permutation σ that minimizes the size of the resulting greedy matching. The key difference from the original problem is **the focus on the adversary’s optimization challenge, rather than the algorithm’s strategy**. The authors establish that approximating this problem within any factor better than $\frac{6}{5}$ is NP-hard, assuming the Small Set Expansion Hypothesis. This result highlights the computational intractability the adversary faces, adding a new dimension to the understanding of the problem’s complexity.

Secondly, to explore the strategic possibilities for both the algorithm and the adversary, the authors introduce a **fractional variant** of the problem. In this variant, items and players are conceptually divided into smaller units called **atoms** allowing for **fractional permutations** that can assign priorities with greater granularity. This differs from the original problem, which only considers integral permutations over whole items and players. A significant finding is that if the algorithm uses only integral permutations, the adversary’s optimal response does not benefit from fractional permutations; an integral player permutation suffices to minimize the resulting matching size. This suggests that fractional strategies do not offer additional advantages to the adversary under these conditions.

Building on this fractional framework, the authors propose the round-robin fractional algorithm. Inspired by the classic Ranking algorithm, this method constructs the priority order over item atoms by cyclically selecting atoms from each item in a round-robin fashion, ensuring a balanced distribution of priorities. The authors prove that this algorithm approaches a competitive ratio of at least $1 - \frac{1}{e}$ when the atom granularity parameter m is sufficiently large. However, they also demonstrate that this ratio is tight for a certain class of regular graphs, indicating limitations of their round-robin algorithm.

1.4.3 Resource Allocation and Pricing Mechanisms

The max-min greedy matching problem is closely related to resource allocation and pricing mechanisms in economics and computer science. In particular, it models scenarios where a seller or platform sets prices or priorities for items, and buyers arrive sequentially, making purchasing decisions based on these prices.

[9] studied the design of pricing mechanisms for allocation in markets, where agents have valuations for items, and the seller sets item prices. In their model, agents arrive adversarially after observing the prices and select items that maximize their utility (value minus price). They showed that for any weighted bipartite graph representing agent-item valuations, there exist item prices that guarantee at least $\frac{1}{2}$ of the optimal welfare, regardless of the agents’ arrival order.

The question arises whether this ratio can be improved when the seller knows the exact valuations in advance. In unit-demand settings with binary valuations (values in $\{0, 1\}$), the max-min greedy matching problem becomes directly relevant, as setting prices corresponds to imposing a priority order over items. Understanding the max-min greedy matching problem can thus lead to better pricing strategies that improve welfare guarantees beyond the $\frac{1}{2}$ threshold [7]. Other works have explored posted price mechanisms and their effectiveness in various combinatorial settings [15, 6, 8].

1.5 Organization

Chapter 2 introduces the necessary notions and definitions for the problem. We reduce the problem to a special class of graphs that have perfect matchings. This allows us to transform an input bipartite graph G into an equivalent directed graph H , which simplifies further analysis. Next, we prove several basic claims about the problem that act as groundwork for the main results of the thesis.

Chapter 3 is devoted to presenting an algorithm that guarantees the ratio of $\frac{5}{9}$. For this result, we build on the idea from [7] about the importance of long directed paths for our problem. Their method can be described as follows: first, they construct a path decomposition P of the input graph. Then, they construct multiple permutations π based on P . Each permutation imposes a lower bound on the competitive ratio, which depends on parameters related to the path decomposition P . They prove that the maximum of these lower bounds exceeds the 0.5 threshold, in particular they show that it equals $\frac{1}{2} + \frac{1}{86}$.

[7] mention that their method is quite involved and ask whether simpler methods exist. We start by describing our simplified version of their approach that achieves a competitive ratio above 0.5. We slightly modify their path decomposition, which allows us to reduce the number of considered permutations π down to three.

Next, we study the parameters of P involved in the lower bound inequalities. We consider the case when all lower bounds are equal, as this corresponds to the minimum possible competitive ratio guarantee. As a result, we modify our path decomposition further. This enables us to reduce the number of considered permutations π down to one.

We investigate the classes of graphs for which our method above achieves the best possible competitive ratio guarantee. For certain graphs, the method provides a guarantee of $\frac{1}{2} + \frac{1}{54}$. At the same time, for a class of Hamiltonian graphs, our method attains $\frac{5}{9}$ guarantee. This result for Hamiltonian graphs arises from the fact that, in such cases,

the decomposition P consists of a single path. More generally, we observe that a smaller number of paths in the decomposition P leads to a better guarantee.

To obtain $\frac{5}{9}$ guarantee for all graphs, we first generalize $\frac{5}{9}$ guarantee to any decomposition P that consists of paths of even size and at most one path of odd size. We refer to such a decomposition as a **balanced vertex group**. This step is crucial to our work as it serves as a fundamental building block in our method.

Next, we introduce the concept of a **balanced path decomposition**, which is a sequence of balanced vertex groups. We develop a linear time algorithm that constructs a balanced path decomposition P_b for any directed graph H . P_b enjoys key structural properties about edges between vertex groups.

Each balanced vertex group of P_b individually achieves $\frac{5}{9}$ guarantee. However, edges between vertex groups break this guarantee. Therefore, in order to extend this guarantee to the original graph, we leverage structural properties of the balanced path decomposition. In particular, this decomposition prevents the existence of certain types of edges between balanced vertex groups. These structural properties enable us to establish $\frac{5}{9}$ guarantee for arbitrary graphs.

Chapter 4 concludes the thesis by summarizing our findings and discussing potential directions for future research.

Chapter 2

Preliminaries

Throughout the work we will use the following notions and definitions.

General notions

- Unless specified otherwise, G refers to an undirected bipartite graph, H refers to a directed graph.
- For (directed or undirected) graph G , define $V(G)$ as the set of vertices of G , $E(G)$ as the set of edges of G .
- In our work, we will mostly consider directed graphs that contain self-loops (v, v) for each vertex v of H .
- Notation $\{v_1, v_2, \dots, v_k\}$ refers to a set of elements, while (v_1, \dots, v_k) refers to an ordered sequence of elements.
- Define $|G|$ as the size of (directed or undirected) graph G that is equal to the number of vertices of G .
- For a directed graph H , existence of a directed edge from v_i to v_j may be denoted as (v_i, v_j) as well as $(v_i \rightarrow v_j)$.
- A set of vertices O is called an **anticlique** if there are no edges between distinct vertices of O .
- An undirected graph G is called a **bipartite graph** if its set of vertices can be split into two disjoint sets V and U such that V and U are anticliques.

- For a (directed or undirected) graph G on a vertex set V and a subset $U \subseteq V$, we define an **induced subgraph** $G[U]$ of G with the vertex set U and the edge set $\{(v, u) \in E(G) \mid v \in U, u \in U\}$.
- For a directed graph H and a subset of vertices X , define

$$N^-(X) = \{u : (u \rightarrow x) \in H, x \in X\}$$

In this work, we would only consider directed graphs H that contain self-loops (v, v) for every vertex v of H , hence $N^-(X)$ includes X .

- For a directed graph H and two disjoint sets of vertices A and B , we say that there are no edges from A to B if there doesn't exist an edge $(a \rightarrow b)$ with $a \in A, b \in B$. Generally, when we refer to edges between A and B or edges from A to B , we refer to the set of edges (a, b) with $a \in A, b \in B$.
- For a sequence A and a set B , define $A \setminus B$ as a subsequence of A constructed from A by excluding all elements of $A \cap B$.
- For a sequence A and a set B , define $A \cap B$ as a subsequence of A constructed from A by retaining only elements of $A \cap B$.
- For a set A and a sequence B , define $A \setminus B$ as $A \setminus C$, where C is the set of elements of B .

Permutations

- We define permutation $\pi = (\pi_1, \pi_2, \dots, \pi_{|V|})$ of a vertex set V as a **priority ordering** of V .
- We say that $\pi_1 \in V$ has the highest priority, while $\pi_{|V|} \in V$ has the lowest priority.
- When vertex a has higher priority than vertex b , we say that a is **prioritized** over b , or equivalently a is **left** of b as if we draw π left to right.
- We will define the notation of concatenating sequences and sets that would define a permutation. For example we may write $\pi = (S, W)$ with $S = (s_1, \dots, s_k)$ being a sequence, $W = \{w_1, \dots, w_t\}$ being a set. If we use a sequence, it means that the order in π is fixed within that sequence. On the other hand, if we use a set, it means that

we are allowed to choose any order for the elements of this set. For example, writing $\pi = (S, W)$ means that π is allowed to be defined as $\pi = (s_1, \dots, s_k, w_1, \dots, w_k)$ as well as $\pi = (s_1, \dots, s_k, w_k, \dots, w_1)$.

We use this notation to construct π that guarantees a high ratio for the max-min greedy matching problem. Thus our construction and related statements have certain flexibility. In particular, the related statements hold for any choice of π that falls within the above notation.

Matchings

- A **matching** of (directed or undirected) graph G is a subset of edges M where no two edges share a vertex.
- A size of a matching M is defined as $|M|$ and is equal to the number of edges in M .
- $V(M)$ is defined as the set of vertices incident to M . Note that $|V(M)| = 2|M|$.
- A **perfect matching** of a (directed or undirected) graph G is a matching that matches all vertices of G .
- A subset of edges M of a directed graph H is called a **functional** if for each vertex v there is at most one outgoing edge ($v \rightarrow u$) and at most one incoming edge ($u \rightarrow v$) in M . Note that any matching is also a functional.
- For a functional M and vertex set X , define

$$M(X) = \{u : (x \rightarrow u) \in M, x \in X\}$$

- Note that $M(X)$ is defined for any $X \subseteq V$. However, we will only use such notion for X that satisfies $|M(\{x\})| = 1$ for all $x \in X$.
- For a functional M and a vertex x , we define $M(x)$ as the only element of $M(\{x\})$. In this case, $M(x)$ is called a **projection** of x .
- For a directed graph H and matching M , define $start(M) = \{v : (v \rightarrow u) \in M\}$, $end(M) = \{u : (v \rightarrow u) \in M\}$. We say that M **leaves** a set of vertices $X = start(M)$. Similarly, we say that M **enters** a set of vertices $Y = end(M)$.
- For a directed edge $e = (v \rightarrow u)$, define $start(e) = v$, $end(e) = u$.
- Define \overline{m}_G as the size of a maximum matching of an undirected graph G .

Paths

- A sequence of vertices $P = (v_1, \dots, v_k)$ is called a **path** in (directed or undirected) graph G if edge (v_i, v_{i+1}) is present in G for all $1 \leq i < k$.
- In this thesis we only consider paths with distinct vertices.
- In this thesis we only consider paths in directed graphs.
- We define the **length** of P as the number of edges of P , which is equal to $k - 1$, and $|P| = k$ as the **size** of P , that is equal to the number of vertices of P .
- A simple path P is called a **Hamiltonian path** if it contains all vertices of G exactly once. If G contains a Hamiltonian path, G is said to be a **Hamiltonian graph**. Note that in graph theory literature, a Hamiltonian graph usually is a graph that contains a cycle that visits all vertices of a graph exactly once. Unlike this common definition, our definition of a Hamiltonian graph assumes an existence of a path that visits all vertices of a graph exactly once.
- A **cycle** $C = (v_1, \dots, v_k)$ of a graph G is defined as a sequence of vertices such that there exists a path (v_1, \dots, v_k) and an edge (v_k, v_1) .
- Define $start(P)$ as the first vertex of a path P .
- Define $end(P)$ as the last vertex of path P .
- A path P is called a **singleton** path if $|P| = 1$.
- For path P , let $V(P)$ be the set of vertices of P . For a sequence of paths $T = (P_1, \dots, P_k)$, define $V(T) = \cup_i V(P_i)$.

Let G be a bipartite graph on a vertex set $U \cup V$. Let π be a priority ordering of V , σ be a priority ordering of U . Consider the process of going over elements of σ from σ_1 to $\sigma_{|U|}$. For each vertex $u \in \sigma$, match u with yet unmatched vertex $v \in \pi$ that is adjacent to u . If no such v exists, u is left unmatched. If multiple such v exist, take the leftmost v with respect to π .

This greedy process results in a matching $M_G[\pi, \sigma]$. Let us also define $m_G[\pi, \sigma] = |M_G[\pi, \sigma]|$. We call this process the **greedy matching process**, and the obtained matching the **greedy matching**.

Define

$$\begin{aligned}\rho[G, \pi, \sigma] &= \frac{m_G[\pi, \sigma]}{\overline{m}_G} \\ \rho[G, \pi] &= \min_{\sigma} \rho[G, \pi, \sigma] \\ \rho[G] &= \max_{\pi} \rho[G, \pi] = \max_{\pi} \min_{\sigma} \frac{m_G[\pi, \sigma]}{\overline{m}_G}.\end{aligned}$$

Similarly, define

$$\begin{aligned}q[G, \pi, \sigma] &= 1 - \rho[G, \pi, \sigma] \\ q[G, \pi] &= 1 - \rho[G, \pi] \\ q[G] &= 1 - \rho[G].\end{aligned}$$

Note that calculating $\rho[G]$ can be interpreted as a two-player game. The first player chooses π to maximize $m_G[\pi, \sigma]$ without knowing σ in advance. The second player, given π chosen by the first player, chooses σ to minimize $m_G[\pi, \sigma]$.

We are interested in studying the game from the perspective of the first player. For that reason, we would like to know whether the first player can guarantee that $\rho[G]$ is greater than some value ρ_0 for all bipartite graphs G . Define

$$\rho = \min_G \rho[G] \quad \text{and} \quad q = 1 - \rho,$$

where G ranges over all bipartite graphs.

The goal of this work is to study ρ and $\rho[G, \pi]$. From the perspective of the first player, we would like to come up with a strategy \mathcal{P}_x . When \mathcal{P}_x is applied to arbitrary G it produces $\pi = \mathcal{P}_x(G)$ with $\rho[G, \pi] \geq x$. The first player would like to maximize x .

2.1 Initial Observations and Simplifications

First, we reduce the class of graphs to consider. It turns out that it is enough to study graphs that contain a perfect matching.

Lemma 2.1.1 ([2]). *Let \mathcal{G} be the set of bipartite graphs that contain a perfect matching, or equivalently the set of bipartite graphs $G = (V \cup U; E)$ that satisfy $|V| = |U| = \overline{m}_G$. Define $\rho_{\mathcal{G}} = \min_{G \in \mathcal{G}} \rho[G]$. Then we have $\rho_{\mathcal{G}} = \rho$.*

For better understanding, below we describe the sketch of the proof of Lemma 2.1.1.

Let M be a maximum matching in arbitrary bipartite graph G . Consider an induced subgraph $G' = G[V(M)]$ on a vertex set $V(M) = V' \cup U'$. Note that G' contains a perfect matching M . Suppose we were able to construct a priority ordering π' on V' that achieves a competitive ratio ρ' on G' , or in other words $\rho[G', \pi'] = \rho'$.

Consider an induced subgraph $G'' = G[V' \cup U]$. Then $\rho[G'', \pi'] \geq \rho[G', \pi'] = \rho'$. To see this, notice that expanding set U' to set U will not decrease the number of matched vertices in V' , as every additional vertex in $U \setminus U'$ can only match an additional vertex in V' that wasn't matched before.

Finally, consider $\pi = (\pi', V \setminus V')$ and an arbitrary priority ordering σ on U . Since V' precedes other vertices of V in π , it implies that the greedy matching process applied to G and π, σ will exactly mirror the greedy matching process applied to an induced subgraph $G[V' \cup U]$ and π', σ . Then $\rho[G, \pi] \geq \rho[G'', \pi'] \geq \rho'$. Therefore, π achieves competitive ratio of at least ρ' for G as any π' does for G' , which proves the Lemma 2.1.1.

From now on, we may assume that $|V| = |U| = n$ and that there exists a perfect matching (v_i, u_i) for $1 \leq i \leq n$.

A matching M of a graph G is called **maximal** if it is impossible to obtain a larger matching M' by extending M via inclusion of any edge $e \in E(G) \setminus M$. In other words, $M \cup \{e\}$ is not a matching for any $e \in E(G) \setminus M$. Equivalently, any edge e of G shares a vertex with some edge of M , otherwise we could include e in M , obtaining a larger matching $M' = M \cup \{e\}$.

For the next claim, we need the following well known result. Its proof can be found in [19].

Lemma 2.1.2. *Any maximal matching of G has size at least $\frac{\overline{m}_G}{2}$.*

Proof. Consider a maximum matching M , i.e. a matching M with $|M| = \overline{m}_G$. Consider a maximal matching T with $t = |T|$. Notice that any $e' \in T$ may be adjacent to at most two distinct edges of M . Each edge $e \in M$ must be adjacent to some $e' \in T$, otherwise T is not maximal as it can be extended with e . Therefore it requires at least $\frac{\overline{m}_G}{2}$ edges in T to have an adjacent edge in T for each $e \in M$, which proves the claim. \square

We use the claim from [7] that any greedy matching is actually a maximal matching, which gives us a simple lower bound of $\frac{1}{2}$ on $\rho[G, \pi]$ for any G and π .

Lemma 2.1.3 ([7]). *A greedy matching is a maximal matching.*

Proof. Suppose $M = M_G[\pi, \sigma]$ is a greedy matching that is not maximal. Therefore, there exists an edge $e = (v, u)$ such that $M \cup \{e\}$ is a matching. Consider the moment when vertex u was processed in σ . Since u was left unmatched, it means that all neighbors of u were matched. However, v is a neighbor of u that was eventually left unmatched, contradiction. \square

Lemma 2.1.4 ([7]). *We have $\rho \geq \frac{1}{2}$.*

Proof. From Lemma 2.1.2 and Lemma 2.1.3 it follows that $m_G[\pi, \sigma] \geq \frac{\bar{m}_G}{2}$ for any π, σ . Hence, $\rho[G, \pi] \geq \frac{1}{2}$ and $\rho \geq \frac{1}{2}$. \square

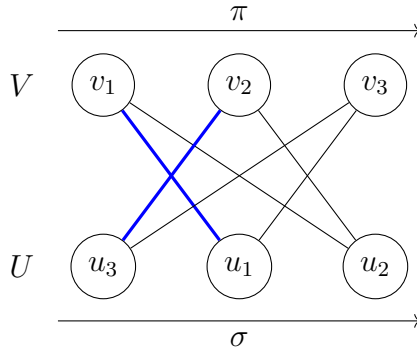


Figure 2.1: Graph G for Lemma 2.1.5

Next, we prove that no strategy can achieve ρ better than $\frac{2}{3}$ by providing a counterexample graph that was introduced in [4].

Lemma 2.1.5 ([4]). *There exists a graph G with $\rho[G] = \frac{2}{3}$.*

Proof. Define G on $V = \{v_1, v_2, v_3\}, U = \{u_1, u_2, u_3\}$, edges of G consist of two perfect matchings $\{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$ and $\{(v_1, u_2), (v_2, u_3), (v_3, u_1)\}$. See Figure 2.1. Consider any π . Using symmetry of G , by renumbering vertices of G we may assume that $\pi_3 = v_3$. Set $\sigma = (u_3, u_1, u_2)$. Let us simulate the greedy matching process for π, σ .

- u_3 is adjacent to v_3, v_2 and will be matched to v_2 .
- u_1 is adjacent to v_3, v_1 and will be matched to v_1 .
- u_2 is adjacent to v_1, v_2 , both of which are already matched, hence u_2 and v_3 remain unmatched.

We get that $m_G[\pi, \sigma] = 2$, hence $\rho[G] \leq \frac{2}{3}$. Note that $\rho[G] \geq \frac{1}{2}$ by Lemma 2.1.4 and $\rho[G]$ is a multiple of $1/3$, hence $\rho[G] = \frac{2}{3}$. \square

Note that we can construct an infinite family of graphs with $\rho[G] = \frac{2}{3}$. To do so, construct a family of bipartite graphs $\{G_k\}$ where G_k consists of k copies of G .

2.2 Reduction to Directed Graph

We would like to implicitly incorporate the assumption of Lemma 2.1.1 that our graph contains a perfect matching. Without loss of generality, suppose that there exists a perfect matching (v_i, u_i) for $1 \leq i \leq n$ in G . Call u_i a **twin** of vertex v_i . Informally, we assume that u_i will be matched to v_i , unless u_i is matched to (or **spoiled** by) some other vertex v_j . In that case v_i may be left unmatched.

We define a directed graph H on a vertex set V , where $(v_i \rightarrow v_j) \in H$ if and only if $(u_i, v_j) \in G$. The graph H is called a **spoiling** graph of G , introduced in [7].

Let us describe how H and G are related in terms of the greedy matching. Fix π . Without loss of generality, let $\pi = (v_1, \dots, v_n)$. Consider any greedy matching $M_G = M_G[\pi, \sigma]$ for some σ , let $m = |M_G|$. There is a bijection between edges of G and H . Therefore, we could consider a bijection $M_H \subseteq E(H)$ of M_G , where M_H is a functional on H . Note that if M_G includes (v_i, u_i) , then M_H includes the loop $(v_i \rightarrow v_i)$. An example relation between M_H and M_G can be observed in Figure 2.2.

In this work, we are mainly interested in the vertices of G that are not matched in M_G . For the sake of space, (unless specified otherwise) when we speak about the set of unmatched vertices in V or in U , we refer to the set of vertices unmatched in M_G .

We say that an edge $(v_j \rightarrow v_i)$ is **oriented left** if $j > i$. Similarly, we say that a matching M in H is **oriented left** if every edge of M is oriented left. The following Lemma shows crucial property of $V \setminus V(M_G)$, i.e. the vertices in V that are unmatched in M_G .

Lemma 2.2.1. *For a greedy matching $M_G = M_G[\pi, \sigma]$, define $X = V \setminus V(M_G)$. Suppose M is a subset of M_H that leaves X , then M is oriented left.*

Proof. In order for $X \subseteq V$ to be unmatched in M_G , twins of X must be matched in M_G . Consider any $v_i \in X$. Its twin u_i must be matched to v_j with $j < i$ and $v_j \notin X$. Otherwise, u_i would have been matched to v_i instead, due to higher priority of v_i in π . This fact corresponds to an edge $(v_i \rightarrow v_j)$ in M_H that is oriented left. \square

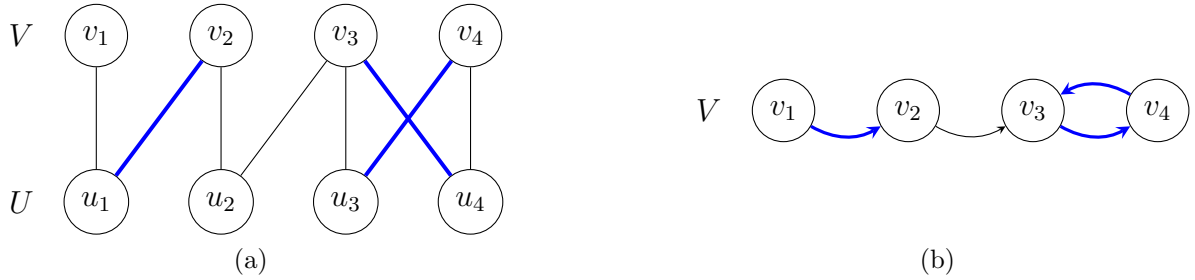


Figure 2.2: (a) Matching M_G in graph G ; (b) The corresponding functional M_H in graph H

Corollary 2.2.2. *If $X \subseteq V$ is the set of vertices such that every edge leaving X is oriented right with respect to π , then X is matched for any σ .*

Proof. Apply Lemma 2.2.1 to every element of X separately. \square

As a corollary, v_1 is matched for any σ . This is because there is no directed edge from v_1 that is oriented left, as there are no vertices to the left of v_1 . To extend this idea further, if there exists π that introduces no edges oriented left, then $\rho[G, \pi] = 1$, because no vertex $v \in V$ can remain unmatched. On the other hand, if no such π exists, then there exists σ that results in at least one unmatched vertex. We formalize these ideas below.

An ordering π of vertices of a directed graph D is called a **topological ordering** if every edge $(\pi_i \rightarrow \pi_j)$ of D satisfies $i \leq j$. In other words, there is an ordering of vertices of D that directs all edges to the right. If a topological ordering exists for D , then D is called a **directed acyclic graph**, or DAG for short. We need the following well known result about topological orderings [5].

Lemma 2.2.3. *There exists a topological ordering π of D if and only if there are no directed cycles in D .*

Proof. Suppose that there are no directed cycles in D . If there is a vertex v without incoming edges to it then we could recursively construct π' on $D[V(D) \setminus \{v\}]$, and define $\pi = (v, \pi')$. Otherwise, every vertex v has an incoming edge. Then, we can find a directed cycle in D . For this, start by taking any v and repeatedly move along some incoming edge. Eventually, we will arrive at some vertex that was visited before, hence there is a directed cycle in D , contradiction.

Suppose that there is a topological ordering π of D . Suppose C is a directed cycle in D that is not a self-loop, let $(v \rightarrow u)$ be any edge on C . C defines a directed path from u

to v . Since v is prioritized over u in π , at least one edge on this path must be directed left with respect to π , contradiction. \square

Topological ordering is very important for max-min greedy matchings. Here is a simple fact that highlights it.

Lemma 2.2.4 ([7]). *If π is a topological ordering of H , then $m_G[\pi, \sigma] = n$ for any σ .*

Proof. It follows from Corollary 2.2.2 that there are no unmatched vertices in H , as there are no edges $(v_i \rightarrow v_j)$ with $j > i$. \square

The following Lemma is a basic building block for most of the results of this thesis.

Lemma 2.2.5 ([7]). *Suppose $\pi = (v_1, \dots, v_n)$. Let $X = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$ be a subset of $V \setminus V(M_G)$, where $i_1 < i_2 < \dots < i_k$. Then there are at least $|N^-(X)|$ vertices in $\{v_1, v_2, \dots, v_{i_k}\} \setminus X$ matched in M_G .*

Proof. For this proof, we consider both an undirected graph G and a corresponding directed graph H interchangeably. Note that G and H share a set of vertices V . The notation $N^-(\cdot)$ is used only for directed graph H .

Let $T \subseteq U$ be the set of vertices adjacent to X in G . Note that $|T|$ is equal to $|N^-(X)|$. By definition of greedy matching, each $u \in T$ must be matched to a vertex in $\{v_1, \dots, v_{i_k-1}\}$, otherwise u is supposed to be matched to its neighbor in X . The claim follows. \square

Consider the greedy matching process for a directed graph H and permutations π, σ . Let $W \subseteq V(H)$ be a subset of vertices of a graph H . Define $\bar{U}_H(W)$ as the set of **unmatched vertices** within W . When the graph H is clear from the context, we simply write $\bar{U}(W)$. Throughout the thesis, we allow ourselves to use $\bar{U}(W)$ in cases, when σ is not explicitly defined. In those cases, we assume that the claim is true for any choice of σ . Generally speaking, if σ is not explicitly defined, but is necessary for the statement, we assume that the statement is true for any choice of σ .

Throughout the thesis, Lemma 2.2.5 is going to be applied in the following way. Consider π that is formed as a concatenation of a sequence of disjoint sets A_1, \dots, A_k in this order, where elements of each set can be arranged arbitrarily. For each set A_i , we define $X_i = \bar{U}(A_i)$. Later using properties of the method of constructing π , we show that

$|N^-(X_i)| \geq \alpha|X_i|$ for some $\alpha > 0$. With that at hand, we write inequalities of the form

$$\alpha|X_i| \leq |N^-(X_i)| \leq \sum_{j=1}^i |A_j| - \sum_{j=1}^i |X_j| \quad (2.1)$$

$$\sum_{j=1}^i |X_j| + \alpha|X_i| \leq \sum_{j=1}^i |A_j| \quad (2.2)$$

where the second inequality of (2.1) holds due to Lemma 2.2.5 applied to X_i . Inequalities of type (2.2) allow us to bound $\sum_{j=1}^k |X_j|$.

Since Lemma 2.2.5 operates on a set of unmatched vertices, it becomes convenient to estimate $|\bar{U}(V)|$ instead of $m_G[\pi, \sigma]$. For simplicity we define $q(\pi) = q[G, \pi]$, assuming that G is implicitly determined from the context.

To rephrase Lemma 2.1.4, it is true that $q(\pi) \leq \frac{1}{2}$ for any π . In terms of q , the goal of the first player is to find π such that $q(\pi)$ is sufficiently small. For the rest of the thesis, when we say bound, we will refer to the upper bound on q or $q(\pi)$, unless specified otherwise.

Chapter 3

Main result

This chapter is devoted to presenting an algorithm that finds π in linear time with respect to the number of vertices and edges of H . This π satisfies $q(\pi) \leq \frac{4}{9}$ or, in other words, guarantees that the ratio of unmatched vertices to the total number of vertices in G is upper bounded by $\frac{4}{9}$.

We start by describing our simplified modification of the method in [7], which achieves a bound below 0.49. To achieve this, we introduce the greedy path decomposition P in Section 3.1.

This decomposition has valuable properties. Namely, each path of P corresponds to a large matching in G , while the decomposition ensures the absence of certain edges between paths. Using these properties, [7] construct several permutations π^i , each resulting in an upper bound $q(\pi^i)$ parameterized by P . [7] prove that the minimum of these bounds is below 0.49. We describe the proof of [7] in Section 3.2.

Next, we study the parameters of P involved in the inequalities. We aim to achieve a stronger relation between them, as it would allow us to strengthen the overall bound. For this reason, we describe a method of transforming the decomposition from Section 3.1. This transformation allows us to impose an additional equality relation between the parameters of P . This transformation is described in Section 3.3.

Using an additional equality derived previously, we construct a single permutation π satisfying $q(\pi) \leq 0.49$. This construction is described in Section 3.4.

Referring to π constructed in Section 3.4, we find that our method cannot prove a guarantee on $q(\pi)$ beyond $\frac{4}{9}$, and for some graphs it only proves $q(\pi) \leq \frac{1}{2} + \frac{1}{54}$. At the same time, our method proves $q(\pi) \leq \frac{4}{9}$ for the class of Hamiltonian graphs of even size.

We present the result of [7] that proves $\frac{4}{9}$ upper bound for Hamiltonian graphs of any size, as described in Section 3.5.

We generalize this result for decomposition P that consists of paths of even size and at most one path of odd size. We refer to such a decomposition as a **balanced vertex group**. We prove $\frac{4}{9}$ upper bound for graphs that are representable as a balanced vertex group.

This result leads us to define the concept of a **balanced path decomposition**, which is a sequence of balanced vertex groups. Using ideas from Section 3.3, we develop a linear time algorithm that constructs a balanced path decomposition P_b for any directed graph H . P_b enjoys key structural properties about edges between vertex groups. This is the crucial part of our work, it is described in Section 3.6.

Using structural properties of the balanced path decomposition P_b , we prove $\frac{4}{9}$ upper bound for arbitrary graphs in Section 3.7.

In conclusion, we discuss potential improvements of our method in Section 3.8.

3.1 Initial Path Decomposition

To begin, we give an intuition on why it may be interesting to study path decompositions for this problem. Consider a special case when H contains Hamiltonian path P . Let $s = \text{start}(P)$ be the starting vertex of P . Partition vertices of $V(P) \setminus \{s\}$ into sets V_e and V_o , where V_e is the set of vertices at even distance from s in P (excluding s itself), and V_o is the set of vertices at odd distance from s in P .

Let M_1 be a matching consisting of edges of P that enter vertices of V_o . Existence of M_1 guarantees $|N^-(V_o)| \geq 2|V_o|$. Hence, we can apply Lemma 2.2.5 to $\bar{U}(V_o)$ and restrict $|\bar{U}(V_o)|$. Similar logic applies to V_e . This approach gives good $q(\pi)$ bound because both V_e and V_o are of large size roughly $\frac{n}{2}$. In other words, we utilize the existence of two large matchings (one enters V_o and the other enters V_e).

We start by describing our simplified version of [7]. We start by describing the greedy path decomposition.

Lemma 3.1.1 (adapted from [7]). *There exists a decomposition of H into a sequence of vertex disjoint directed paths $P = (P_1, P_2, \dots, P_k)$ with the following properties. Let $s_i = \text{start}(P_i)$. Let T be the set of singleton paths of P . Then*

1. For each $1 \leq i \leq k - 1$, there are no directed edges from any vertex of P_{i+1}, \dots, P_k to s_i .
2. T consists of the last $|T|$ paths of P , being $(P_{k-|T|+1} \dots P_k)$.
3. T forms an anticlique.

Proof. We will construct P iteratively, starting with P_1 . Initially define P consisting of the only path P_1 , that consists of a single vertex. Suppose we already constructed paths P_1, P_2, \dots, P_{i-1} that satisfy property 1. Let F be the set of remaining vertices. We maintain the fact that there are no edges from F to any $\{s_1, \dots, s_{i-1}\}$. We show how to construct P_i .

If F is an anticlique, then append $|F|$ singleton paths to P , finishing the construction. Otherwise, start P_i with any edge $(v \rightarrow u)$ that is present in F . Repeatedly extend P_i by prepending any vertex $f \in F$ that satisfies $(f \rightarrow \text{start}(P_i)) \in E(H)$. Suppose that P_i cannot be extended, let $s_i = \text{start}(P_i)$. Notice that no vertex from F has an edge to s_i . Append P_i to P , notice that no vertex of P_i has an outgoing edge to any vertex s_j with $j < i$. Moreover, there is no edge from any vertex of F to any vertex of $\{s_1, \dots, s_i\}$. Continue with construction of P_{i+1} . \square

We derive an important corollary that will be necessary for our construction of permutations π .

Corollary 3.1.2. *For decomposition P defined in Lemma 3.1.1, sequence $S = (s_1, \dots, s_k)$ forms a topological ordering.*

Proof. Follows from property 1 in Lemma 3.1.1. \square

3.2 $\frac{1}{2} - \varepsilon$ Bound

Consider path decomposition $P = (P_1, \dots, P_k)$ from Lemma 3.1.1. Define $s_i = \text{start}(P_i)$, $S = (s_1, \dots, s_k)$, $R = V \setminus S$, $r = |R| = n - k$. Let V_e (V_o) be a subset of $V(P) \setminus S$ such that elements of V_e (V_o) are at even (odd) distance from S in their corresponding path in P , excluding vertices of S . Consider only edges from S to R , let M_{sr} be a maximum matching among those edges, define $m = |M_{sr}|$.

Below we describe a slight modification of the result of [7] and prove that $q[G] \leq 0.49$ for any G . Understanding this proof is useful for understanding key methods and concepts that will be used throughout the work.

Lemma 3.2.1 (adapted from [7]). Define $\pi^1 = (s_1, s_2, \dots, s_k, V_e, V_o)$. Then $q(\pi^1) \leq \frac{4}{9} + \frac{k}{9n}$.

Lemma 3.2.2 (adapted from [7]). Define $\pi^2 = (S, R)$. Then $q(\pi^2) \leq \frac{1}{3} + \frac{r}{3n} - \frac{m}{3n}$.

Lemma 3.2.3 (adapted from [7]). Define $\pi^3 = (R, S)$. Then $q(\pi^3) \leq \frac{m}{2n} + \frac{r}{2n}$.

Combining all three, we get the following result.

Lemma 3.2.4 (adapted from [7]). One of $\{\pi_1, \pi_2, \pi_3\}$ defined in Lemmas 3.2.1, 3.2.2, 3.2.3 satisfies $q(\pi_i) \leq 0.49$.

The rest of this Section is devoted to proving this result. First, we need the following statement.

Lemma 3.2.5. *There exists a matching that enters V_e , and there exists a matching that enters V_o .*

Proof. Consider the set of edges that precedes V_e (V_o) in P . It forms the desired matching. \square

Proof of Lemma 3.2.1. Corollaries 3.1.2 and 2.2.2 imply that all vertices of S are matched. Let $X = \bar{U}(V_e)$, $Y = \bar{U}(V_o)$, $x = |X|$, $y = |Y|$. We have $q(\pi^1) = \frac{x+y}{n}$. Lemma 3.2.5 implies $|N^-(X)| \geq 2x$ and $|N^-(Y)| \geq 2y$. Applying Lemma 2.2.5 to X , we have $2x$ matched vertices in $V_e \cup S$ and x unmatched vertices in V_e . This results in $3x$ unique vertices in $V_e \cup S$. Therefore, we have

$$3x \leq |V_e| + k.$$

Applying Lemma 2.2.5 to Y , there are $2y$ matched vertices along with $x+y$ unmatched vertices in V . Therefore,

$$x + 3y \leq n.$$

Summing up, we get

$$2 \cdot (3x) + 3 \cdot (x + 3y) \leq 2(|V_e| + k) + 3 \cdot (n).$$

Notice $|V_e| \leq |V_o|$, since each vertex of V_e is preceded by a vertex in V_o . Thus,

$$2(|V_e| + k) + 3n \leq (k + |V_e| + |V_o|) + k + 3n = 4n + k$$

and thus we have

$$2 \cdot (3x) + 3 \cdot (x + 3y) = 9x + 9y \leq 4n + k$$

which results in $q(\pi^1) = \frac{x+y}{n} \leq \frac{4}{9} + \frac{k}{9n}$. \square

Proof of Lemma 3.2.2. Similar to the proof of Lemma 3.2.1, all vertices of S are matched. Note that $|N^-(R)|$ is at least $m + |R|$ due to existence of M_{sr} . Let $X = \bar{U}(R)$, $x = |X|$. Observe that

$$|N^-(X)| = |N^-(R \setminus (R \setminus X))| \geq |N^-(R)| - 2 \cdot |R \setminus X|$$

where the last inequality holds because each vertex in R contributes at most 2 to $|N^-(R)|$. This is because each vertex v of R can be assigned to at most 2 distinct vertices in $N^-(R)$, those are being v itself and its adjacent vertex in M_{sr} , in case v is matched in M_{sr} . Then

$$|N^-(X)| \geq |N^-(R)| - 2 \cdot |R \setminus X| \geq m + |R| - 2|R| + 2x = m - r + 2x.$$

Applying, Lemma 2.2.5, the number of matched vertices in graph is at least $|N^-(X)| = m - r + 2x$. Given that there are x unmatched vertices, we derive

$$n \geq (m - r + 2x) + x = m - r + 3x$$

$$3x \leq n + r - m$$

$$q(\pi^2) = \frac{x}{n} \leq \frac{1}{3} + \frac{r}{3n} - \frac{m}{3n}$$

as desired. □

Proof of Lemma 3.2.3. Suppose $X = \bar{U}(R)$, $Y = \bar{U}(S)$, $x = |X|$, $y = |Y|$. The only edges that leave vertices of S and are oriented left in π are edges that enter R , since S is a topological ordering. Therefore, the only edges that leave vertices of $Y \subseteq S$ and are oriented left in π are edges that enter R . Hence, Lemma 2.2.1 implies $y \leq |M_{sr}| = m$. Moreover, it implies that the only edges that leave vertices of $X \cup Y$ and are oriented left must enter R . Therefore, Lemma 2.2.5 implies

$$\begin{aligned} x + y &\leq |N^-(X \cup Y)| \leq |R \setminus X| = r - x \\ 2x + y &\leq r \end{aligned}$$

Combining with inequality $y \leq m$, we get

$$\begin{aligned} 2x + 2y &\leq r + m \\ q(\pi^3) = \frac{x + y}{n} &\leq \frac{m}{2n} + \frac{r}{2n}. \end{aligned}$$

□

Proof of Lemma 3.2.4. We have

$$\begin{aligned} q[G] &\leq q(\pi^1) \leq \frac{4}{9} + \frac{k}{9n} \\ q[G] &\leq q(\pi^2) \leq \frac{1}{3} + \frac{r}{3n} - \frac{m}{3n} \\ q[G] &\leq q(\pi^3) \leq \frac{m}{2n} + \frac{r}{2n} \end{aligned}$$

Taking weighted average of inequalities with weights

$$\frac{18}{23}, \frac{3}{23}, \frac{2}{23}$$

we get

$$q[G] \leq \frac{2k + 2r}{23n} + \frac{9}{23} = \frac{11}{23} \leq 0.49$$

which concludes the proof. \square

3.3 Tightening the Decomposition

Recall that m is defined as the size of a maximum matching among directed edges from S to $V \setminus S$. Let us focus our attention on m . Specifically, let us explain what facts about m can be deduced from properties of decomposition P .

Let t be the number of singleton paths P_i in P . By construction, it is necessary that those are the last t paths P_{k-t+1}, \dots, P_k . Note that m is at least $k - t$, as there is a matching that leaves (s_1, \dots, s_{k-t}) consisting of the first edge of each non-singleton path (P_1, \dots, P_{k-t}) .

This fact does not help in improving the bound for q . However, we could tighten the inequality on m by reconstructing P to get $m = k - t'$, where t' is the number of singleton vertices in the reconstructed version of P . This would allow us to achieve $q(\pi) \leq 0.49$ using a single permutation instead of three.

Lemma 3.3.1. *It is possible to construct a path decomposition $P = (P_1, \dots, P_k)$ of H with the following properties. Define $s_i = \text{start}(P_i)$, $S = (s_1, \dots, s_k)$, $R = V \setminus S$. Let M_{sr} be a maximum matching among directed edges from S to R , $m = |M_{sr}|$. Let T be the set of singleton paths in P . Define $t = |T|$. Then*

- T consists of the last $|T|$ paths of P , namely $(P_{k-t+1} \dots P_k)$.
- T forms an anticlique.
- $S = (s_1, \dots, s_k)$ is a topological ordering.
- m is equal to the number of non-singleton paths in P , which is $k - t$.

First, we derive an important corollary that we will utilize later for proving $q(\pi) \leq 0.49$ for a single permutation π .

Corollary 3.3.2. *Let P be a path decomposition defined in Lemma 3.3.1. Let f_i be the vertex that succeeds s_i on a non-singleton path P_i for each non-singleton P_i , let F be the set of all f_i . Then, every edge from T to $(V \setminus T) \setminus S$ must be between T and F .*

Proof. If there is an edge $t \rightarrow u$ with $t \in T$ and $u \in ((V \setminus T) \setminus S) \setminus F$, then we could construct a matching $(s_1 \rightarrow f_1), (s_2 \rightarrow f_2), \dots, (s_{k-t} \rightarrow f_{k-t}), (t \rightarrow u)$ that has size $k - t + 1$, which violates assumption $m = k - t$. \square

Corollary 3.3.2 actually hints at how such P may be constructed. If a violation edge $(t \rightarrow u)$ is found, we can reconstruct P similarly to how augmenting paths are used in Kuhn's algorithm [14]. This method is formally described below.

Proof of Lemma 3.3.1. Start by constructing a decomposition P defined in Lemma 3.1.1. Let f_i be the vertex that succeeds s_i on a non-singleton path P_i for each non-singleton P_i . Let F be the set of all f_i .

Suppose there exists an edge $(s_i \rightarrow u)$ that participates in M_{sr} for $s_i \in S \setminus T$. If s_i is not matched to f_i in M_{sr} and f_i isn't matched in M_{sr} , reconstruct M_{sr} by including edge $(s_i \rightarrow f_i)$ in M_{sr} instead of $(s_i \rightarrow u)$. We repeat this operation for each s_i and f_i that satisfy conditions above. The process is finite since the number of (s_i, f_i) edges in M_{sr} is increased each time. We call this a **preprocessing phase**.

After applying the preprocessing phase, we assume that if s_i is matched in M_{sr} , then f_i is also matched in M_{sr} .

Suppose $m > k - t$. Then, there must exist an edge $(s_i \rightarrow u_1)$ in M_{sr} for some $u_1 \in R \setminus F$ because $|F| = k - t$ and $|M_{sr}| = m > k - t$. We will construct a sequence A via the process below.

- If there exists an edge $(v \rightarrow u_1) \in M_{sr}$ satisfying $v \in T$ and $u_1 \in R \setminus F$, then we define $A = (v, u_1)$ and finish.
- Otherwise, there must exist an edge $(s_{i_2} \rightarrow u_1) \in M_{sr}$ with $s_{i_2} \notin T$ and $u_1 \in R \setminus F$. Then f_{i_2} must be matched in M_{sr} due to preprocessing phase. Suppose f_{i_2} is matched to s_{i_3} in M_{sr} .
- If $s_{i_3} \in T$, then we define $A = (s_{i_3}, f_{i_2}, u_1)$ and finish.
- Otherwise, we repeat the same chain of thought for s_{i_3} and f_{i_3} and so on. At some point, f_{i_g} would be matched to some $v \in T$ in M_{sr} because $|M_{sr}| > |S \setminus T|$. In that case, we define $A = (v, f_{i_g}, f_{i_{g-1}}, \dots, f_{i_2}, u_1)$ and finish.

As a result, we define A that has the form $(v, f_{i_g}, f_{i_{g-1}}, \dots, f_{i_2}, u_1)$, where $v \in T$ and $u_1 \in R \setminus F$. Suppose that v is a singleton vertex of path $P_{i_{g+1}}$. Suppose that u_1 belongs to $V(P_{i_1})$. By construction of A , M_{sr} contains $(v \rightarrow f_{i_g}), (s_{i_g} \rightarrow f_{i_{g-1}}), \dots, (s_{i_2} \rightarrow u_1)$.

Next, we define the transformation of P . Informally, reconstruct P by setting edges $(v \rightarrow f_{i_g}), (s_{i_g} \rightarrow f_{i_{g-1}}), \dots, (s_{i_2} \rightarrow u_1)$ as the first edges of paths $P_{i_{g+1}}, \dots, P_{i_2}$, respectively. In addition, cut P_{i_1} between u_1 and its predecessor in P_{i_1} , shortening P_{i_1} and allowing for suffix of P_{i_1} that starts with u_1 to be attached to another path.

Formally, for $2 \leq j \leq g$, let $Q_{i_j} = P_{i_j} \setminus \{s_{i_j}\}$ be a suffix of path P_{i_j} that excludes the first vertex of P_{i_j} . Let Q_{i_1} be the suffix of path P_{i_1} starting from vertex u_1 . Let L be the prefix of the path P_{i_1} up until vertex u_1 , excluding u_1 .

Set $P_{i_j}^{new} = (s_{i_j}, Q_{i_{j-1}})$ for $2 \leq j \leq g+1$, and set $P_{i_1}^{new} = L$. It is essential that u_1 doesn't belong to F , so that $P_{i_1}^{new}$ doesn't become a singleton path. For the rest of paths, set $P_i^{new} = P_i$. Finally, since T is an anticlique, we can exchange $P_{i_{g+1}}^{new}$ with P_{k-t+1}^{new} , preserving the fact that all singleton paths remain on a suffix and the fact that S remains a topological ordering. We declare $P^{new} = (P_1^{new}, P_2^{new}, \dots, P_k^{new})$ as a new version of P .

By applying this transformation to P , we have decreased the number of singleton paths in P , while preserving necessary conditions on T and S . By repeatedly applying this transformation, at some point we would not be able to find a sequence A which would imply $m = s - t$. \square

3.4 $\frac{1}{2} - \varepsilon$ Bound Using a Single Permutation

The goal of this section is to derive $q(\pi) \leq 0.49$ for some explicitly constructed permutation π . To achieve this, we study the permutation π^1 from Lemma 3.2.1.

Let us first provide the intuition behind our approach. Recall the proof of Lemma 3.2.1. We derived the following inequalities:

$$3x \leq |V_e| + k \quad (3.1)$$

$$x + 3y \leq n \quad (3.2)$$

$$x + y \leq \frac{4}{9}n + \frac{k}{9} \quad (3.3)$$

$$q(\pi^1) = \frac{x + y}{n} \leq \frac{4}{9} + \frac{k}{9n} \quad (3.4)$$

We wish to strengthen inequality (3.4) to derive $q(\pi^1) \leq \frac{1}{2} - \varepsilon$ for $\varepsilon > 0$. Suppose that inequality (3.4) does not guarantee any bound on $q(\pi^1)$ better than $\frac{1}{2}$. In that case, it is necessary that k is at least $\frac{n}{2}$ due to inequality (3.4). On the other hand, $k > \frac{n}{2}$ implies $|\bar{U}(V)| = x + y \leq |V \setminus S| = n - k < \frac{n}{2}$. Therefore, k must be asymptotically equal to $\frac{n}{2}$, otherwise $q(\pi^1)$ could be upper bounded by some $\frac{1}{2} - \varepsilon$ for $\varepsilon > 0$.

We would like to upper bound $|V_e| + k$ in inequality (3.1), it would allow us to improve the upper bound for $q(\pi^1)$. Given that k is fixed as $\frac{n}{2}$, deriving an upper bound for $|V_e| + k$ is the same as deriving an upper bound for $|V_e|$. Consider two extreme cases.

- Suppose $k = \frac{n}{2}, t = 0$. Then, P must consist of $\frac{n}{2}$ paths of size 2. It implies that $|V_o|$ is equal to k and $|V_e|$ is equal to 0, which in turn implies $3x \leq |V_e| + k \leq k = \frac{n}{2}$. After reflecting this fact in inequality (3.1), it could be derived that $q(\pi^1)$ is at most 0.49.
- Suppose $k = \frac{n}{2}, t = k - 1$. In this case, we have only one non-singleton path P_1 of length $n - k = \frac{n}{2}$, in which case $|V_o|$ and $|V_e|$ are roughly equal to $\frac{n}{4}$. In this case, the inequality (3.1) is replaced by $3x \leq \frac{3}{4}n$. In such case, the inequality (3.4) only results in $q(\pi^1) \leq \frac{1}{2}$.

In two extreme cases that we considered, we have $|V_e| = 0$ when $t = 0$ and $|V_e| = \frac{n}{4}$ when $t = \frac{n}{2} - 1$. These two cases suggest that smaller t may result in a better upper bound for $|V_e|$. We formalize this idea in the following lemma.

Lemma 3.4.1. *Let P be a decomposition from Lemma 3.3.1. Then,*

$$|V_e| + k - t \leq \frac{2}{3}n.$$

Proof. Each element of V_e is preceded by an element of V_o in P ; hence, $|V_e| \leq |V_o|$. Each element of s_1, \dots, s_{k-t} is succeeded by an element of V_o ; hence, $k-t \leq |V_o|$. Therefore,

$$n-t = k + |V_e| + |V_o| - t = (k-t) + |V_e| + |V_o| \leq 3|V_o|$$

which implies

$$\frac{n-t}{3} \leq |V_o|.$$

Finally,

$$\begin{aligned} |V_e| + k - t &= (k + |V_e|) - t = (n - |V_o|) - t = n - t - |V_o| \leq \\ &\leq n - t - \frac{n-t}{3} = \frac{2}{3}(n-t) \leq \frac{2}{3}n. \end{aligned}$$

□

The last step is to replace $|V_e| + k$ with $|V_e| + k - t$ in inequality (3.1). Recall that S is placed at the beginning of π^1 so that all elements of S remain matched. Using Corollary 3.3.2, we could move T between V_e and V_o , which does not introduce unmatched vertices in T but improves the bound in inequality (3.1).

Lemma 3.4.2. *Let P be a decomposition defined in Lemma 3.3.1.*

Consider $\pi = (s_1, s_2, \dots, s_{k-t}, V_e, T, V_o)$. Then, we have

$$q(\pi) \leq \frac{1}{2} - \frac{1}{54}.$$

Proof. The sequence (s_1, \dots, s_{k-t}) forms a topological ordering; hence, it does not contain any unmatched vertices. Applying Corollary 3.3.2 and the fact $F \subseteq V_o$, there exist no left edges that leave T . Applying Corollary 2.2.2 to T , T does not contain any unmatched vertices.

Define $X = \bar{U}(V_e)$, $Y = \bar{U}(V_o)$, $x = |X|$, and $y = |Y|$. We have $q(\pi) = \frac{x+y}{n}$. Applying Lemma 2.2.5 to X , there are $2x$ matched vertices in $V_e \cup (S \setminus T)$ and x unmatched vertices in V_e . This results in $3x$ unique vertices in $V_e \cup (S \setminus T)$. Applying Lemma 3.4.1, we derive

$$3x \leq |V_e| + k - t \leq \frac{2}{3}n. \tag{3.5}$$

Furthermore, there are $2y$ matched vertices along with $x+y$ unmatched vertices in V . Therefore,

$$x + 3y \leq n. \tag{3.6}$$

Adding 2 times inequality (3.5) and 3 times inequality (3.6), we obtain

$$2 \cdot 3x + 3(x + 3y) = 6x + 3x + 9y = 9x + 9y \leq \frac{4}{3}n + 3n = \frac{13}{3}n.$$

Therefore,

$$x + y \leq \frac{13}{27}n = \left(\frac{1}{2} - \frac{1}{54}\right)n.$$

□

3.5 Special Case of Hamiltonian Graphs

Refining the term $|V_e| + k - t$ in inequality (3.5) would directly improve the upper bound on $q(\pi)$. Let us focus our attention on the $k - t$ term.

Consider the best possible case of $k - t = 1$. Specifically, suppose that P consists of a single Hamiltonian path, then $k = 1, t = 0$. If n is odd, then $|V_e| + k = \frac{n+1}{2}$ and $|V_o| = \frac{n-1}{2}$, which results in $q(\pi) \leq \frac{4n+1}{9n}$. This is insufficient to achieve the desired result of $q(\pi) \leq \frac{4}{9}$.

Therefore, the first step is to provide a solution for the case of Hamiltonian graphs. We adapt the method proposed in [7].

Lemma 3.5.1 (Adapted from [7]). *Given a Hamiltonian graph H , there exists a permutation π such that $q(\pi) \leq \frac{4}{9}$.*

Proof. We adapt the proof of Lemma 3.2.1. Consider the permutation $\pi = (s_1, V_e, V_o)$ with P as the Hamiltonian path; then $k = 1$ and $t = 0$. Define $x = |\bar{U}(V_e)|$ and $y = |\bar{U}(V_o)|$. Suppose that n is even; then $|V_e| + k = \frac{n}{2}$, we obtain

$$\begin{aligned} 3x &\leq \frac{n}{2} \\ x + 3y &\leq n \\ x + y &\leq \frac{2 \cdot \frac{n}{2} + 3n}{9} = \frac{4}{9}n. \end{aligned}$$

Now, suppose that $n = 2m + 1$ is odd. Then

$$3x \leq |V_e| + k = m + 1. \tag{3.7}$$

If $m + 1$ is not divisible by 3, then the inequality (3.7) can be strengthened to $3x \leq m \leq \frac{n}{2}$. Similar to the case of even n , we derive the same result $x + y \leq \frac{4}{9}n$.

Otherwise, suppose $m + 1$ is divisible by 3. Let $z = \text{end}(P)$ be the last vertex of P ; note that $z \in V_e$. Consider the permutation $\pi' = (s_1, V_e \setminus \{z\}, V_o, z)$. Define $X = \bar{U}(V_e \setminus \{z\})$, $Y = \bar{U}(V_o \cup \{z\})$. Define $x = |X|$, $y = |Y|$.

Note that $m - 2$ is the largest number divisible by 3 that is less than or equal to m . Thus, we strengthen $3x \leq m$ to $3x \leq m - 2$.

Observe that $|N^-(Y)|$ is at least $2|Y| - 1$. To prove it, notice that every vertex of $Y \setminus \{z\} \subseteq V_o$ has a distinct predecessor in P that belongs to $\{s_1\} \cup (V_e \setminus \{z\})$. The only exception to this rule for elements of Y is z , in case z belongs to Y . Therefore, $x + 3y - 1 \leq n$. Overall, we obtain

$$\begin{aligned} 3x &\leq m - 2 \\ x + 3y &\leq n + 1. \end{aligned}$$

Summing up, we get

$$\begin{aligned} 6x + 3(x + 3y) &\leq 2m - 4 + 3n + 3 = 4n - 2 \\ x + y &\leq \frac{4n - 2}{9} \leq \frac{4}{9}n. \end{aligned}$$

Therefore, $q(\pi') \leq \frac{4}{9}$. □

3.6 Balanced Path Decomposition

As we have seen, the proof of Lemma 3.5.1 is more complicated for the case when n is odd. This suggests that constructing a path decomposition with as many even-length paths as possible may be advantageous, ideally with all paths being of even length. However, when n is odd, it is not possible to have all paths of even length; in such cases, allowing one path to have an odd length is acceptable. To formalize this idea, we define the following special type of path decomposition.

A **balanced vertex group** is a decomposition of V into directed paths P_1, P_2, \dots, P_k , with $s_i = \text{start}(P_i)$, that satisfies the following properties:

- $S = (s_1, \dots, s_k)$ is a topological ordering.

- For each $1 \leq i \leq k - 1$, path P_i has an even number of vertices.
- The last path P_k may have any number of vertices.

Lemma 3.6.1. *Given a balanced vertex group $C = (P_1, \dots, P_k)$, there exists a permutation π such that $q(\pi) \leq \frac{4}{9}$.*

Proof. Let V_e (V_o) be a subset of $V(C) \setminus S$ such that elements of V_e (V_o) are at even (odd) distance from S in their corresponding path in C , excluding vertices of S . Define $z = \text{end}(P_k)$.

If n is even, then $|V_e| + k = \frac{n}{2}$; otherwise, $|V_e| + k = m + 1$ for $n = 2m + 1$. The rest of the proof is identical to the proof of Lemma 3.5.1. \square

To construct π that achieves $q(\pi) \leq \frac{4}{9}$ for an arbitrary graph H , we will employ the following approach. We decompose V into a sequence of balanced vertex groups. Each group, when considered as an induced subgraph, has at most $\frac{4}{9}$ fraction of vertices unmatched. It remains to handle edges between groups that may cause more vertices to be unmatched. We informally describe the process below.

We employ a greedy approach to construct a maximal balanced vertex group. At each step, the current path P_k is attempted to be extended, as in Lemma 3.1.1. If this step fails, then there is an option to start the next path P_{k+1} not from a single vertex but from a suffix of P_k . It would allow us to ensure that the size of P_k is even. However, this option requires a certain condition to hold, namely the condition on line 17 of Algorithm 1.

If the condition on line 17 is false, then P_k may have odd size, hence we are forced to declare (P_1, \dots, P_k) as the balanced vertex group and restart the process on the remaining vertex set F . We are forced to restart since allowing more than one odd-size path is not permitted.

However, the fact that the condition on line 17 is false implies that certain edges between F and P_k do not exist. This fact allows us to combine the $\frac{4}{9}$ bounds of independent balanced vertex groups. We proceed with describing the formal proof.

Algorithm 1: Iterative Balanced Path Decomposition

Input: Directed graph $H = (V, E)$
Output: Decomposition \mathcal{C}

- 1 Initialize $F \leftarrow V, \mathcal{C} \leftarrow \emptyset, C \leftarrow \emptyset$
- 2 Define $C = (P_1, P_2, \dots, P_k)$, where each P_i is a path
- 3 Assume $P_k = (s_k, o_1, e_1, o_2, e_2, \dots, z_k)$, where $s_k = \text{start}(P_k)$ and $z_k = \text{end}(P_k)$
- 4 **while** *True* **do**
- 5 **if** $F = \emptyset$ **then**
- 6 $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
- 7 **Return** \mathcal{C}
- 8 **else if** $C = \emptyset$ **then**
- 9 Choose any $v \in F$
- 10 $F \leftarrow F \setminus \{v\}$
- 11 $P_1 \leftarrow (v)$
- 12 $C \leftarrow (P_1)$
- 13 **else if** $\exists f \in F \mid (f \rightarrow s_k) \in E$ **then**
- 14 Choose such f
- 15 $F \leftarrow F \setminus \{f\}$
- 16 Prepend f to P_k
- 17 **else if** $\exists f \in F, \exists e_i \in P_k \mid (f \rightarrow e_i) \in E$ **then**
- 18 Let $e_i \in E(P_k)$ be the vertex that is the closest to s_k in P_k that satisfies
- 19 $\exists f \in F \mid (f \rightarrow e_i) \in E$
- 20 Choose corresponding f, e_i
- 21 $P_k^{\text{new}} \leftarrow (s_k, o_1, e_1, \dots, o_i)$
- 22 $P_{k+1} \leftarrow (f, e_i, o_{i+1}, e_{i+1}, \dots, z_k)$
- 23 $C \leftarrow (P_1, \dots, P_{k-1}, P_k^{\text{new}}, P_{k+1})$
- 24 **else**
- 25 $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
- 26 $C \leftarrow \emptyset$

Define the following notation for a balanced vertex group C :

- Define $S(C) = (s_1, \dots, s_k)$.
- Let $E(C)$ be a subset of $V(C) \setminus S(C)$ such that elements of $E(C)$ are at even distance from $S(C)$ in their corresponding path in C .

- Let $O(C)$ be a subset of $V(C) \setminus S(C)$ such that elements of $O(C)$ are at odd distance from $S(C)$ in their corresponding path in C .
- Define $L(C) = E(C) \cup S(C)$.
- Define $end(C) = end(P_k)$.

Sets $E(P)$, $O(P)$, and $L(P)$ are defined similarly for a single path P .

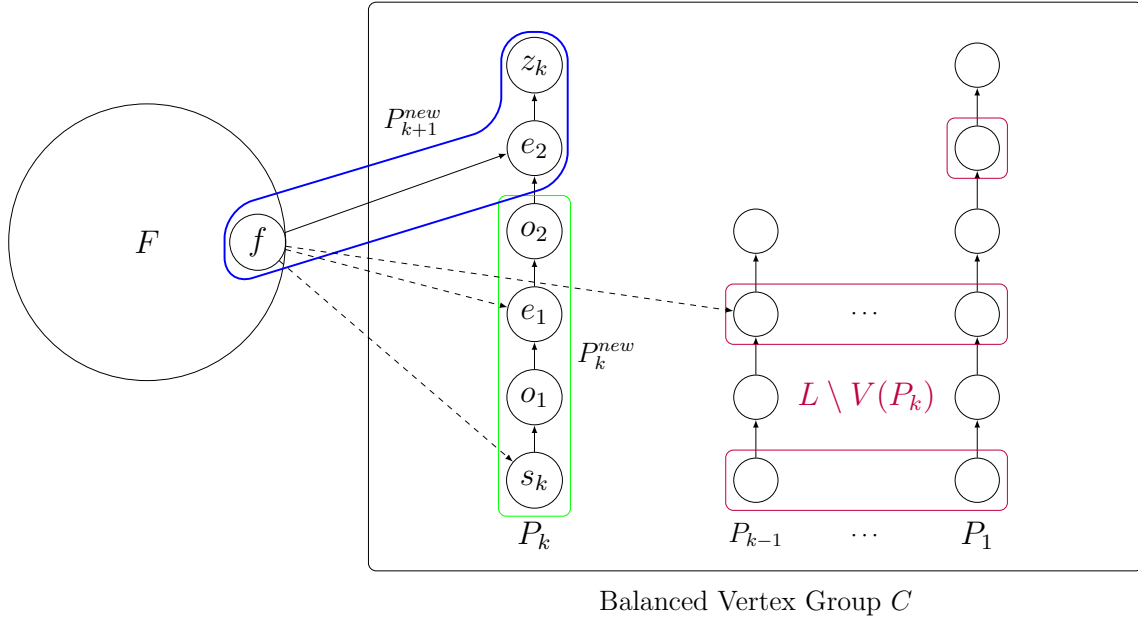


Figure 3.1: Visualization of step 4 in Lemma 3.6.2. Dashed lines represent edges that are guaranteed not to exist in a graph

Lemma 3.6.2. *Algorithm 1 outputs a vertex decomposition of directed graph H into balanced vertex groups C_1, \dots, C_g such that there are no directed edges from any vertex of C_j to any vertex of $L_i = L(C_i)$ for all $i < j$.*

Proof. We will construct this decomposition iteratively. Define a set of unassigned vertices F , a set of groups constructed so far $\mathcal{C} = (C_1, C_2, \dots)$, and a current group $C = (P_1, P_2, \dots, P_k)$. Note that C is excluded from \mathcal{C} . Define $L_i = L(C_i)$, $L = L(C)$. Assume $P_k = (s_k, o_1, e_1, o_2, e_2, \dots, z_k)$. For simplicity, z_k may be referred to as e_i or o_i for some i , depending on the length of P_k and the parity of $|P_k|$. The following properties are maintained:

- There is no directed edge from any vertex of C_j to any vertex of L_i for all $i < j$.
- There is no directed edge from any $f \in F \cup C$ to any $l \in L_i$ for all i .
- There is no directed edge from any $f \in F$ to any $l \in L \setminus V(P_k)$, where P_k is the last path in C .
- All elements of \mathcal{C} , as well as C , remain a balanced vertex group.

Initially, set $F = V$, $\mathcal{C} = \emptyset$, and $C = \emptyset$ (line 1). At each step, exactly one of the following operations is performed in the specified order, whichever applies first:

1. (line 5) Suppose F is empty. Append C to \mathcal{C} and terminate the process.
2. (line 8) Otherwise, F is nonempty. Suppose C is empty. Move any element f of F into P_1 and remove f from F .
3. (line 13) Otherwise, both F and C are nonempty. Suppose there exists $f \in F$ satisfying $(f \rightarrow s_k) \in E(H)$. Prepend f to P_k and remove f from F .
4. (line 17) Otherwise, suppose there exists $f \in F$ satisfying $(f \rightarrow e_i) \in E(H)$ for some i . Find such e_i with the smallest possible i . Set $P_k^{\text{new}} = (s_k, o_1, e_1, \dots, o_i)$ and $P_{k+1}^{\text{new}} = (f, e_i, o_{i+1}, e_{i+1}, \dots, z_k)$. Set $F^{\text{new}} = F \setminus \{f\}$. For better understanding, refer to Figure 3.1.

Since e_i is the closest vertex to s_k among $E(P_k)$ with an incoming edge from F , there does not exist a directed edge from F^{new} to $L(P_k^{\text{new}})$. Note that P_k^{new} has even size, so C remains a balanced vertex group.

5. (line 24) Otherwise, there are no directed edges from F to L . Append C to \mathcal{C} and set $C^{\text{new}} = \emptyset$.

After each step, either $|F|$ decreases or $|\mathcal{C}|$ increases. Therefore, at the end of the process, the desired decomposition is obtained. \square

Lemma 3.6.3. *Algorithm 1 runs in $O(|V| + |E|)$ time.*

Proof. Hash set (described in [5]) is a data structure to store a set of elements. Hash set supports the following operations in constant time on average:

- Insertion of an element.

- Deletion of an element.
- Checking if a given element x exists in a set.
- Retrieving any element that exists in a set. No guarantees are given for the retrieved element, only the fact that it was present in a set.

The set F is maintained as a hash set, allowing extraction of an element from F for step 2 and removal of elements from F . For each vertex u , maintain $set(u) = N^-(u) \cap F$ at all times. The collection \mathcal{C} is stored in a list, as elements are only appended to \mathcal{C} .

Steps 1 and 2 are trivial to implement in constant time. For steps 3 and 4, at each step we iterate over vertices s_k, e_1, e_2, \dots of P_k in this order. When considering vertex x , check in constant time whether there exists $f \in F$ satisfying $(f \rightarrow x) \in E(H)$ by verifying if $set(x)$ is not empty. If such f is found, proceed accordingly; otherwise, the vertex x will not be considered again in steps 3 and 4. This is because x moves to P_k^{new} , and only vertices from the last path of \mathcal{C} are considered. When removing f from F , iterate over vertices $u \in N^+(f)$ and remove f from $set(u)$. \square

3.7 Proving $\frac{4}{9}$ Bound

The final step is to combine balanced vertex groups constructed in Lemma 3.6.2 into a single permutation π . Consider a decomposition into groups C_1, \dots, C_g . Define n_i as the number of vertices in C_i . Consider a subgraph H_i of H induced on $V(C_i)$. Apply Lemma 3.6.1 on H_i . For simplicity, assume that Lemma 3.6.1 constructs permutation $\pi_i = (S(C_i), E(C_i), O(C_i))$. Define $x_i = |\bar{U}_{H_i}(E(C_i))|$ and $y_i = |\bar{U}_{H_i}(O(C_i))|$. Note that $x_i + y_i \leq \frac{4}{9}n_i$.

A naive approach for constructing π for H would concatenate permutations π_i into $\pi = (\pi_1, \pi_2, \dots, \pi_g)$ in an attempt to achieve $\sum_{i=1}^g x_i + y_i \leq \sum_{i=1}^g \frac{4}{9}n_i = \frac{4}{9}n$. However, sets $\bar{U}_H(E(C_i))$ or $\bar{U}_H(O(C_i))$ may be larger than their counterparts in H_i due to existence of edges from vertices of C_i to vertices of C_j for all $j < i$.

To overcome this issue, we apply an argument similar to the one about singleton paths in Lemma 3.4.2. No edges exist from C_j to $L(C_i)$ for all $i < j$. Therefore, placing $L(C_i)$ before C_j in π preserves the estimate of $|\bar{U}_H(C_j)|$. This suggests constructing π by placing $L(C_i)$ for all i at the beginning of π , followed by $V(C_i) \setminus L(C_i) = O(C_i)$ for all i .

Lemma 3.7.1. *For any directed graph H , there exists permutation π such that $q(\pi) \leq \frac{4}{9}$. Moreover, π can be explicitly constructed in linear time with respect to the number of vertices and edges of H .*

Proof. Construct decomposition $\mathcal{C} = C_1, \dots, C_g$ defined in Lemma 3.6.2. Define n_i as the number of vertices in C_i . Define $z_i = \text{end}(C_i)$, $S_i = S(C_i)$, $O_i = O(C_i)$, $E_i = E(C_i)$, $l_i = \lfloor \frac{n_i}{2} \rfloor$.

We call group C_i **bad** if n_i is odd and $l_i + 1$ is divisible by 3.

- If C_i is bad, set $A_i = (S_i, E_i \setminus \{z_i\})$, $B_i = (O_i \cup \{z_i\})$.
- Otherwise, set $A_i = (S_i, E_i)$, $B_i = O_i$.

Set $\pi = (A_1, A_2, \dots, A_g, B_1, B_2, \dots, B_g)$. Define Z as the set of vertices z_i over all bad C_i , and define $d = |Z|$. Define $X_i = \bar{U}(A_i)$ and $x_i = |X_i|$. By Lemma 3.6.2, there are no directed edges from A_i to A_j for any $j < i$. Therefore set $S_i \cap A_i$ is matched and $X_i \subseteq (A_i \cap E_i)$.

Each vertex v of X_i has a distinct predecessor $u \in O_i$ in corresponding path of C_i . Lemma 3.6.2 implies that there are no directed edges from C_i to any A_j for all $j < i$. Given that both v and u belong to C_i , applying Lemma 2.2.5 to X_i implies that there are at least $2x$ matched vertices in A_i . Hence $3x_i \leq |A_i|$.

For all $1 \leq i \leq g$

- Suppose n_i is even. Then $3x_i \leq |A_i| = l_i$ holds.
- Suppose n_i is odd and C_i is not bad. Then $l_i + 1$ is not divisible by 3, therefore inequality $3x_i \leq |A_i| = l_i + 1$ can be strengthened to achieve $3x_i \leq l_i$.
- Suppose n_i is odd and C_i is bad. Then $l_i + 1$ is divisible by 3, therefore $l_i - 2$ is the largest number not greater than l_i that is divisible by 3. This implies that inequality $3x_i \leq |A_i| = l_i$ can be strengthened to achieve $3x_i \leq l_i - 2$.

Summing inequalities over all i yields

$$\sum_{i=1}^g 3x_i \leq \sum_{i=1}^g l_i - 2d. \quad (3.8)$$

Define $Y = \bar{U}(\cup_i B_i)$. Each element $y \in (Y \setminus Z)$ has a distinct predecessor u in \mathcal{C} belonging to $\cup_i A_i$. Applying Lemma 2.2.5 to Y , $V(H)$ contains at least $|Y| + |Y \setminus Z|$ matched vertices and exactly $\sum_{i=1}^g x_i + |Y|$ unmatched vertices. Summing up terms, this yields

$$\sum_{i=1}^g x_i + 2|Y| + |Y \setminus Z| = \sum_{i=1}^g x_i + 3|Y| - |Y \cap Z| \leq n \quad (3.9)$$

$$\sum_{i=1}^g x_i + 3|Y| \leq n + |Y \cap Z| \leq n + |Z| = n + d. \quad (3.10)$$

Combining two times inequality (3.8) and three times inequality (3.10) gives

$$9\left(\sum_{i=1}^g x_i + |Y|\right) \leq 2\left(\sum_{i=1}^g l_i - 2d\right) + 3(n + d) \leq 4n - d$$

$$\sum_{i=1}^g x_i + |Y| \leq \frac{4}{9}n.$$

Therefore $q(\pi) \leq \frac{4}{9}$. □

3.8 Further Directions

Now that our proof of the $\frac{4}{9}$ bound is complete, let us discuss its potential extensions and improvements.

3.8.1 Imposing an Order Within Sets

The first fundamental direction is to improve the $\frac{4}{9}$ bound. Note that even in the simplest case when the balanced group decomposition consists of a single group, which in turn consists of a single path, we still do not know any result better than $\frac{4}{9}$. Therefore, let us reconsider the case of a Hamiltonian graph H again.

Recall that permutation $\pi = (s, V_e, V_o)$ achieves $q(\pi) \leq \frac{4}{9}$ for the case when n is even. Note that no particular order is imposed on vertices within V_e or V_o . The proof only utilizes information about edges between sets $\{s\}$, V_e , and V_o . Therefore, a natural approach is to impose an order on vertices within V_e and V_o while preserving the relative order of these sets in π .

Consider recursively constructing permutations $\pi_{H[V_o]}$ and $\pi_{H[V_e]}$ for graphs $H[V_o]$ and $H[V_e]$ using the same method that was applied when constructing permutation π for H . Define π' that is constructed from π by imposing an order $\pi_{H[V_o]}$ ($\pi_{H[V_e]}$) on vertices of V_o (V_e) within π . It is possible to improve the upper bound on $q(\pi')$ beyond $\frac{4}{9}$ in a restricted setting. We describe it below.

For a graph H , consider splitting $V = V(H)$ into two disjoint subsets (A, B) . Define a split (A, B) as **balanced** if for any subsets $X \subseteq A, Y \subseteq B$, inequalities $|N^-(X) \cap B| \geq |X|$ and $|N^-(Y) \cap A| \geq |Y|$ hold.

Lemma 3.8.1. *A balanced split exists if and only if H can be decomposed into a sequence of disjoint even length cycles.*

Proof. Suppose H is decomposed into a sequence of disjoint even cycles. For each cycle C , color vertices of C alternately in red and blue. Place red vertices in A and blue vertices in B . The statement follows.

Conversely, suppose there exists a balanced split (A, B) . Apply Hall's marriage theorem [11] separately to the edges directed from A to B and from B to A . This results in directed perfect matchings M_1 from A to B and M_2 from B to A . Define I as a subgraph of H defined on $V(H)$ and consisting of $M_1 \cup M_2$. Each vertex in I has exactly one incoming and one outgoing edge, therefore I consists of a set of vertex disjoint cycles. Each cycle alternates between vertices of A and B , therefore all cycles have even length. Thus I defines a desired decomposition. \square

Define $\pi = (A, B)$. Note that existence of a balanced split is a strong condition. However, constructed π is similar in its properties to the one in Lemma 3.5.1. We believe that proving a claim for a graph that admits a balanced split is analogous to proving the same claim for Hamiltonian graph, except that it allows us to avoid dealing with parity and divisibility inherent to our previous proofs. The following lemma proves an upper bound on $q(\pi)$ that is better than $\frac{4}{9}$. In order to achieve it, we apply the proof of Lemma 3.5.1 recursively with depth 2.

Lemma 3.8.2. *Suppose that H contains a balanced split (L, C) , and $H[L]$ contains a balanced split (A, B) . Then permutation $\pi = (A, B, C)$ satisfies $q(\pi) \leq \frac{7}{16} < \frac{4}{9}$.*

Proof. Note $|A| = |B| = \frac{n}{4}$ and $|C| = \frac{n}{2}$. Define $X = \bar{U}(A)$, $Y = \bar{U}(B)$, $Z = \bar{U}(C)$. Define $x = |X|$, $y = |Y|$, $z = |Z|$.

Note that $|N^-(X)|$ is at least $3|x|$ since $|N^-(X) \cap A| \geq |X|$, $|N^-(X) \cap B| \geq |X|$ and $|N^-(X) \cap C| \geq |X|$. Applying Lemma 2.2.5 to X , this implies existence of $3x$ matched and x unmatched vertices in A .

Note that $|N^-(X \cup Y) \cap C|$ is at least $|X \cup Y|$. Also $|N^-(Y) \cap (A \cup B)| \geq 2|Y|$. Applying Lemma 2.2.5 to $X \cup Y$, there are $x + y + 2y = x + 3y$ matched and $x + y$ unmatched vertices in $A \cup B$.

Finally, $|N^-(Z)| \geq 2|Z|$. Applying Lemma 2.2.5 to Z yields $2z$ matched and $x + y + z$ unmatched vertices in V .

Combining all three observations yields:

$$\begin{aligned} (x) + (3x) &= 4x \leq |A| = \frac{n}{4} \\ (x + y) + (x + 3y) &= 2x + 4y \leq |A| + |B| = \frac{n}{2} \\ (x + y + z) + (2z) &= x + y + 3z \leq n \\ 4(4x) + 8(2x + 4y) + 16(x + y + 3z) &\leq 4\frac{n}{4} + 8\frac{n}{2} + 16n \\ 48(x + y + z) &\leq 21n \\ x + y + z &\leq \frac{7}{16}n. \end{aligned}$$

Therefore $q(\pi) \leq \frac{7}{16}$. □

We suspect that this approach may be extended to Theorem 3.7.1, but requires careful handling of various cases.

3.8.2 Bidirected Graphs

Throughout the work we have been dealing with directed graphs. It may be of independent interest to study certain subclasses of directed graphs to prove stronger bounds that do not apply in the general setting. One such subclass is bidirected graphs.

Define a directed graph H as **bidirected** if for every edge $(v \rightarrow u)$ in H there exists an edge $(u \rightarrow v)$. Note that bidirected graphs can be interpreted as undirected graphs, as

both allow traversing edges between vertices (v, u) in both directions. However, in bidirected graphs these are formally two separate edges. Undirected graphs are better studied compared to directed graphs, hence this may be a promising direction for improvements.

Let us show results for bidirected graphs that are analogous to similar results for directed graphs. First, let us establish an upper bound that holds for any $q(\pi)$ for all bidirected graphs, similar to how $q(\pi) \leq \frac{1}{2}$ holds for all π and all directed graphs.

Lemma 3.8.3. *For a bidirected graph H , any permutation π satisfies $q(\pi) \leq \frac{1}{3}$.*

Proof. Define $X = \bar{U}(V)$. Let M_G be greedy matching, and let M_H be its bijection in H . Define $Y = M_H(X)$. Note that $X \cup Y$ is a subset of $N^-(X)$, therefore $|N^-(X)| \geq 2|X|$. Applying Lemma 2.2.5 to X yields $3|X| \leq n$, therefore $|X| \leq \frac{1}{3}n$. \square

Recall from Lemma 2.1.5 that $\frac{1}{3}$ is a lower bound on $q(\pi)$ for all π in certain directed graphs. Compared to bidirected graphs, $\frac{1}{3}$ is an upper bound on $q(\pi)$ for all π in all bidirected graphs.

Finally, let us show that there exists a bidirected graph H_b such that $q(\pi)$ is at least $\frac{1}{4}$ for all π . This result is similar to the result of Lemma 2.1.5.

Lemma 3.8.4. *There exists a bidirected graph H such that any permutation π satisfies $q(\pi) \geq \frac{1}{4}$.*

Proof. Define $H = C_4$ as a bidirected cycle of length 4 with vertices $V = (v_1, v_2, v_3, v_4)$ in order of appearance on the cycle. Let u_i be twin of v_i for all $1 \leq i \leq 4$. Let π be any permutation of V . Without loss of generality, suppose that π_4 is equal to v_1 and that v_2 is prioritized over v_4 in π .

Define $\sigma = (u_1, u_2, u_4, u_3)$. Consider the greedy matching process for π, σ .

- Vertex u_1 is adjacent to v_2, v_4, v_1 , and will be matched to v_2 .
- Vertex u_2 is adjacent to v_2, v_3, v_1 . Since v_2 is matched and v_1 has the lowest priority in π , u_2 will be matched to v_3 .
- Vertex u_4 is adjacent to v_3, v_4, v_1 . Since v_3 is matched and v_1 has the lowest priority in π , u_4 will be matched to v_4 .
- Vertex u_3 is adjacent to v_2, v_3, v_4 , all of which are matched. Hence, u_3 and v_1 remain unmatched.

Therefore $m_G[\pi, \sigma] = 3$ and $q(\pi) \geq \frac{1}{4}$. \square

Chapter 4

Conclusion and Further Directions

As the main result of this work, in Chapter 3 we prove that for any graph H , there exists a permutation π with $q(\pi) \leq \frac{4}{9}$. Moreover, this π can be constructed in linear time relative to the number of edges and vertices of the input graph. A gap remains between the lower bound of $\frac{1}{3}$, achieved via a simple graph of size 6 (Lemma 2.1.5), and the upper bound of $\frac{4}{9}$ obtained through our balanced path decomposition.

Recall that there exist large subsegments of π with no internal ordering imposed on them. We suspect that this fact can be exploited to further refine the balanced path decomposition; initial steps are outlined in Section 3.8.1. Moreover, it may be worthwhile to study the problem in the context of bidirected graphs (see Section 3.8.2), as significantly more results are known for undirected graphs compared to directed graphs.

Notice that the graph H depends on which perfect matching of G is chosen. Therefore, another possible direction of research is studying whether the proof can be improved in case a different perfect matching is chosen which results in a different directed graph H .

References

- [1] Allan Borodin, Christos Karavasilis, and Denis Pankratov. An experimental study of algorithms for online bipartite matching. *Journal of Experimental Algorithmics*, 25:1–37, 2020.
- [2] T-H Hubert Chan, Zhihao Gavin Tang, and Quan Xue. Max-min greedy matching problem: Hardness for the adversary and fractional variant. *Theoretical Computer Science*, 986:114329, 2024.
- [3] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 960–979, 2018.
- [4] Vincent Cohen-Addad, Alon Eden, Michal Feldman, and Amos Fiat. The invisible hand of dynamic market pricing. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 383–400, 2016.
- [5] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
- [6] Paul Dütting, Michal Feldman, Thomas Kesselheim, and Brendan Lucier. Prophet inequalities made easy: Stochastic optimization by pricing non-stochastic inputs. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 540–551, 2017.
- [7] Alon Eden, Uriel Feige, and Michal Feldman. Max-min greedy matching. *Theory of Computing*, 18(1):1–33, 2022.
- [8] Tomer Ezra, Michal Feldman, Tim Roughgarden, and Warut Suksompong. Pricing multi-unit markets. *ACM Transactions on Economics and Computation*, 7(4):20:1–20:29, 2020.

- [9] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 123–135, 2015.
- [10] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 982–991, 2008.
- [11] Philip Hall. On representatives of subsets. *Classic Papers in Combinatorics*, pages 58–62, 1987.
- [12] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 587–596, 2011.
- [13] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.
- [14] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [15] Brendan Lucier. An economic view of prophet inequalities. *SIGecom Exchanges*, 16(1):24–47, 2017.
- [16] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.
- [17] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.
- [18] Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science*, 8(4):265–368, 2013.
- [19] Vijay V Vazirani. Approximation algorithms, 2001.