1 Introduction

In the thesis we focus on an algorithmic approach to one of many problems related to resource allocation. More precisely, its main result is a procedure for maintaining a maximum cardinality matching in a setting, where a part of the graph is revealed online, during the algorithm run.

A matching $M$ in a graph $G = \langle V, E \rangle$ is any subset of edges $M \subseteq E$ that are pairwise vertex-disjoint, i.e., each vertex has at most one incident edge in $M$. For example, consider a group of people interconnected by a symmetric relation of acquaintance. A matching then is a pairing of these people such that each person in a pair is acquainted with the other and nobody is in a more than one pair. We call $M$ a maximum cardinality matching, or simply a maximum matching, if it is of biggest possible size, that is, for any other matching $M'$ we have $|M| \geq |M'|$.

It is easiest to see the task of calculating maximum matching as a resource allocation problem in the bipartite case, that is, when the vertices $V$ of $G$ can be partitioned into two sets such that all the edges connect vertices from two different parts. Despite this limitation, practical applications of bipartite matching are numerous, especially, when each edge is labeled by weight that represents the benefit or cost associated with it. For example, medical students in the United States have been assigned to hospitals using a similar setting since the early 50' of the last century. Even before, minimum weight matching was used to optimize problems motivated by transportation or classification of military personnel. Furthermore, both weighted and unweighted versions of the maximum matching problem found multiple uses as building blocks of more complex algorithms, having applications in mobile sensing systems, recommendation systems, algorithms including pattern recognition and several areas of bioinformatics, as well as in
theoretical research on scheduling and load balancing, shortest paths algorithms and hashing.

Over the years, as the research in combinatorial optimization progressed, the notion of matching became one of the fundamental concepts in graph theory. Today it has many different versions and flavors, generalizations and special cases. The subject of the thesis, that is, the problem of maintaining a maximum cardinality matching in an incremental setting, is one such specific variant, which can be summarized as follows. Consider a number of incoming continuous service requests and a known pool of handlers of different abilities—perhaps teams of people, each group able to take care of one job that falls into its area of expertise, or maybe just a farm of servers, each taking care of one request within its capabilities range. Our goal is to continue to maintain as many requests as possible, at the same time minimizing the number of reassignments between the teams and their jobs or between servers and clients connected to them. As argued in [5], possible applications of this problem include streaming content delivery, data storage, job scheduling and hashing.

That setting is an example from a wide class of problems, in which we assume that the input is revealed gradually as the algorithm runs, rather than the all data being given as a whole from the start. Thanks to the increase of easily available computational power, the online approach became not only interesting from theoretical perspective, but also practical on a massive scale. For example, these days, when visiting a website that uses one of the big ad content providers, during the time in which the browser loads the page, huge server farms perform multiple small auctions that decide which vendors will be granted ad space in that particular page impression. Nevertheless, even this vast computing power is not enough to deal with ever-growing amount of information to process—the current monthly internet traffic, as well as the size of some larger databases are measured in exabytes. That forces us to consider algorithms of sublinear efficiency, but how it is possible for the algorithm to run without reading all the data? In a dynamic approach we leverage the fact that large data sets often differ very little from turn to turn, i.e., it is possible to make the procedure run faster by reusing some of the calculation it did on previous rounds.

The online and dynamic approaches are two perspectives on procedures that process data arriving during the algorithm run. The difference between the two depends on the context in which we do the analysis. If we focus on the quality of the solution and decisions made by the algorithm, then we would say it is an online problem, while in a dynamic problem we concentrate on the efficiency of handling the updates.

A perfect example of an online problem is the one studied by Karp, Vazirani
and Vazirani in [16], a work that has hundreds of citations and spurred a whole line of academic research. The input is a bipartite graph $G = \langle U \cup V, E \rangle$ where the vertices of $V$ arrive online, each $v \in V$ being revealed at the same time with all its incident edges. When this happens, the algorithm has to make a decision to leave $v$ unmatched or pair it up with some adjacent vertex $u \in U$ which is still free. Once the choice is made, it is irrevocable, i.e., the matched vertices cannot change their pairs and the unmatched vertices of $V$ have to stay unmatched until the end. The objective is to maximize the size of the resulting matching, and the authors propose an algorithm that has $(1 - \frac{1}{e})$ competitive ratio, which means that the size of the computed solution is at least $(1 - \frac{1}{e})$ times the size of the optimal offline matching. In other words, they analyze the quality of the output by comparing it to the optimal offline solution calculated on the whole final graph $G$.

On the other hand, in the fully dynamic bipartite matching problem we consider the efficiency of the algorithm per single update. In that setting the input graph $G$ is given as a series of both insertions and deletions of edges intermixed with each other, so it is possible for one edge to be added and removed multiple times. The challenge is to develop an algorithm that maintains the maximum cardinality matching and has the fastest possible update times. The online approach does not apply, because each time we calculate the best solution available. Conversely, the dynamic approach does not suit the previous problem, because we are not concerned with efficiency, only how good the final solution will be.

Nevertheless, the distinction between online and dynamic algorithms is not sharply defined. For example, the problem studied in the thesis has characteristics of both—it can be thought of as a middle ground between the two above settings. In a graph-theoretical language, we can formulate it as follows. We are given a bipartite graph $G = \langle W \cup B, E \rangle$ in a one-sided online fashion, that is, each black vertex $b \in B$ arrives together with all its incident edges, exactly as in the work of Karp, Vazirani and Vazirani. However, similarly to the dynamic problem, our goal is to maintain a maximum cardinality matching. In particular, the decisions of the algorithm can be changed later—each time a black vertex $b \in B$ is revealed, the algorithm picks its white pair in $W$, potentially reassigning some other vertices in the process. The problem can be considered dynamic, because we try to calculate, in an efficient manner, an optimal matching for each turn. Even so, in a resemblance to the online setting, we still want to keep the number of rematchings low. In other words, the fewer reallocations are necessary, the better the quality of our solution.

Matching problems are far from being the only problems in the online and dynamic settings. Still, the steadily growing body of research on this topic and the number of papers accepted by the best algorithmic conferences, like IEEE’s
Foundations of Computer Science [19, 6, 1, 9, 18, 22, 12] and ACM’s Symposium on Theory of Computing [16, 21, 15, 17, 20], is an evidence of its importance for the algorithmic graph theory. To put the thesis into a context, in the next section we will sketch advances of selected related papers. For more details, see the Related Work section of the thesis.

2 Related Work

The exact setting we are working with, namely the problem of online bipartite matching with augmentations, was introduced by Grove, Kao, Krishnan and Vitter in [10]. The authors used competitive analysis to show $O(\log |V|)$ bound when each client connects to at most two servers.

That result was then extended by Chaudhuri, Daskalakis, Kleinberg and Lin, who proved $O(|V| \log |V|)$ bound for some restricted models, including forests or random graphs with degree $\Theta(\log |V|)$. They also showed $O(|V| \log |V|)$ bound with high probability for the shortest augmenting path algorithm when the clients arrive in random order.

These are the only works that consider a model that is the same as ours. Nevertheless, there are three general lines of research that are related to the topic of the thesis: online matching algorithms, dynamic matching algorithms and load balancing problems.

When approaching our setting from the load balancing perspective, one can view it as a number of servers waiting for tasks to be assigned—when a new task arrives, before we schedule it for being run on some machine, we might need to migrate some jobs to other servers to accommodate the load. The closest problems are related to online load balancing of permanent tasks with preemption.

Just recently, recently Gupta, Kumar and Stein [11] considered a setting that allows the capacity constrains to be exceeded by a constant factor. They designed an algorithm that maintains such a constant-factor load using only a constant number of reassignments per vertex in the amortized sense.

However, as similar as that model and load balancing problems in general might seem, the results are not quite applicable to our setting. This is because in our model we assume hard capacity constraints, whereas in the load balancing models one usually assumes that the capacities are soft, i.e., the capacities can be exceeded and one is interested in minimizing the maximum load. Hence, these results have somewhat different objectives.

In the context of dynamic matching algorithms it is important to note, that the fully-dynamic model is different than the one analyzed in the thesis. Not only it considers edge updates, but it allows their insertions and deletions to be mixed.
Such a problem is analyzed by Onak and Rubinfeld [21], who presented a constant approximation algorithm that works in polilogarithmic time, a work that was a year later superseded by Baswana, Gupta and Sen [1, 2], who obtained $O(\log |V|)$ update time and factor two approximation.

More recently there were two results with essentially same running time: papers by Neiman and Solomon [20] from STOC and Gupta and Peng [12] from FOCS. The former provides $3/2$-approximation in $O(|E|^{1/2})$, while the latter achieves $(1 + \epsilon)$-approximation in $O(|E|^{1/2} \cdot \epsilon^{-2})$ time.

The problem of online bipartite matching, where vertices from one side arrive online together with all their edges, was introduced in a paper by Karp, Vazirani and Vazirani [16]. The authors apply the framework of competitive analysis and show an algorithm that has $(1 - 1/e)$ approximation factor.

Although the model might seem restricted, actually it has numerous applications, mainly because it is useful to model two sided markets—at the same time it captures some non-trivial characteristics, and yet is simple enough to allow an in-depth analysis. While it is rare that we know both the supply and the demand, often it is not unreasonable to assume that one of them can be reasonably assessed. For example, it might be hard for a hosting company to accurately predict the incoming daily traffic, but it can easily estimate the load that its servers can withstand.

Thus, in 2005 Metha, Saberi, Vazirani and Vazirani [19] generalized the online bipartite matching to accommodate online advertising and present worst-case $(1 - 1/e)$-approximation for this special setting, when budgets of advertisers are large.

The natural barrier of $(1 - 1/e) \approx 0.63$ was “beaten” in the case of the i.i.d. model by Feldman, Mehta, Mirrokni and Muthukrishnan [6].

A similar, but different problem was investigated by Poloczek and Szegedy [22]—a middle ground between online and randomized offline settings. The authors show that the approximation factor of the greedy algorithm is strictly better than $1/2$.

The same year, another modification was considered by Goel and Tripathi, who analyzed a case where the graph is revealed as it is accessed, as if the algorithm had its “eyes closed”. They presented a 0.56 upper bound for their greedy algorithm and 0.7916 lower bound if the procedure always matches just discovered edges. The setting of the third work was inspired by practical application, namely the problem of user conversion in online advertising.

This selection of papers paints, in broad strokes, the landscape of research areas most relevant to this work. Now that the context is known, we move on to present the results included in the thesis.
3 Results

In the thesis we consider the setting of online bipartite matchings with augmentations, introduced by Grove, Kao, Krishnan and Vitter in [10]. Shortly, we are given a bipartite graph $G = (W \cup B, E)$ with its vertices partitioned into static white set $W$ and black set $B$, that is revealed online in a one-sided vertex-incremental fashion. In other words, the set of white vertices $W$ is fixed and assumed to be known beforehand, while the vertices of $B$ arrive one per turn, each with all its incident edges. Our aim is to maintain the maximum cardinality matching, i.e., to be able to output at any given time an optimal matching in a graph that was revealed up to this point. Also, as far as it is possible, we would like to keep the number of reassignments of the vertices low.

We approach this problem using two greedy strategies, both based on the classical augmenting paths technique. The first one is presented in Chapter 2, and the main result is Theorem 2.6 (see section Shortest Paths Approach), which had appeared in “Shortest Augmenting Paths for Online Matchings on Trees” at WAOA’15 [4].

**Theorem 2.6.** If the input graph $G$ is a tree, then the total length of all the augmenting paths applied by the shortest augmenting path algorithm is $O(|V| \log^2 |V|)$.

The other strategy spans Chapters 3–5, and was published at FOCS’14 within a paper titled “Online Bipartite Matching in Offline Time” [3]. It is based on the following heuristic. For each vertex $w \in W$ we kept track of its rank, that is, how many augmenting paths have used $w$. Then, our algorithm uses augmenting paths that are tiered, i.e., paths that greedily minimize the maximum of these counters. In other words, rather than following the shortest paths, we prefer augmenting paths that avoid frequently traversed vertices. The most significant results of Chapter 3 are Theorem 3.16 and its relaxed version Theorem 3.36. For Chapter 4 these are Theorem 4.3 and 4.5 that correspond to Algorithms 3 and 4 respectively. Finally, Chapter 5 provides examples which show that the upper bounds of Theorems 3.16 and 4.3 are tight.

**Theorem 3.16.** For any dynamic unweighted matching algorithm that uses tiered augmenting paths it holds that $\text{rank}(w) \in O(|V|^{1/2})$ for any turn and every white vertex $w \in W$.

**Theorem 4.3.** There exists an algorithm that maintains an exact maximum cardinality matching for the online bipartite matching problem with augmentations in $O(|E| \cdot |V|^{1/2})$ total time.

**Theorem 4.5.** There exists an algorithm that maintains a $(1 - \varepsilon)$-approximate maximum cardinality matching for the online bipartite matching problem with augmentations in $O(|E| \cdot \varepsilon^{-1})$ total time.
It is worth mentioning that prior to [3], the best bounds known for the general case were $O(|V|^2)$ and $O(|E| \cdot |V|)$, respectively, for the total length of augmenting paths and the running time, which corresponds to the naïve algorithm that repeatedly applies just any augmenting path. We now give a more detailed description of results of the thesis.

**Shortest Paths Approach** The first strategy, described in Chapter 2, uses the shortest augmenting paths. Chaudhuri, Daskalakis, Kleinberg and Lin [5] conjecture that such a method achieves $O(|V| \log |V|)$ upper bound on the total number of assignments and reassignments. To this end, in Section 2.1 of the thesis we present a reduction that may potentially simplify any dynamic analysis—the original graph $G$ is transformed into a problem instance in which only vertices of degree 1 are added. This way, the whole structure of $G$ is static and available from the very first turn.

Then, in Section 2.2, we consider the special case when $G$ is a tree and prove the bound of $O(|V| \log^2 |V|)$ on the total length of augmenting paths. That proof is special, because it does not rely on the actual matching it augments each turn, only on the structure of the whole graph. This was possible, thanks to how the concept of minimum surplus was defined in Section 1.2.2—although ideas based on Hall’s surplus are known and appear in literature in various forms, the author has never seen the formulation used in Definition 1.10 of the thesis.

Therefore, in Section 2.3 we propose a setting in which the adversary can change the calculated matching at every turn, in any way that preserves its cardinality. We conjecture that this intervention does not affect the asymptotic length of all the augmenting paths produced by the shortest augmenting paths algorithm.

**Conjecture 2.14.** The total length of all the shortest augmenting paths in the online bipartite matching problem with augmentations is bounded from above by $O(|V| \log |V|)$, even if the adversary is allowed to change the calculated matching at every turn, in any way that preserves its cardinality.

**Ranks and Tiers** The second strategy we apply to the online bipartite matching with augmentations is based on another heuristic. For each vertex we keep count of how many times it was used by augmenting paths, which we call a rank, and try to minimize the maximum of these values over all the vertices. Chapter 3 provides definitions and the theoretical foundations of the approach. Then, we use them in Chapter 4 to design an efficient algorithm that maintains a maximum cardinality bipartite matching in the online setting with augmentations. Chapter 5 describes a number of examples that demonstrate the performance of the technique and
provide corresponding lower bounds on running time of the algorithms and the total length of applied augmenting paths.

The two most important notions of the second strategy are the rank mentioned above and a related concept of tier, both defined in Section 3.1. Section 3.2 provides their most important properties, while Section 3.3 contains the main theoretical result of the thesis, Theorem 3.16. Finally, in Section 3.5 we relax the definitions a bit, so that they are easier to use in an algorithmic setting. In particular, Theorem 3.36 restates Theorem 3.16 using the new relaxed rank.

**Theorem 3.16.** For any dynamic unweighted matching algorithm that uses tiered augmenting paths it holds that \( \text{rank}(w) \in O(|V|^{1/2}) \) for any turn and every white vertex \( w \in W \).

The above theorem allows us to design a number of algorithms for the online bipartite matching problem with augmentations and its different flavors, which we all present in Chapter 4. We begin with Algorithm 1, the search procedure which constitutes the heart of all the other algorithms. Section 4.2 describes two results, which are the main practical contribution of the work, namely Algorithm 3 and Algorithm 4.

Both procedures work in \( O(R \cdot |E|) \) total time and generate augmenting paths of \( O(R \cdot |V|) \) total length, where \( R \) is the maximum achieved rank. The first algorithm maintains an exact maximum cardinality matching, due to Theorems 3.16 and 3.36 its worst-case total running time is bounded from above by \( O(|E| \cdot |V|^{1/2}) \). For the same reason the sum of lengths of augmenting paths it produces is \( O(|V|^{3/2}) \). The second algorithm is parametrized by a positive \( \varepsilon > 0 \) and maintains a \( (1 - \varepsilon) \)-approximation of the maximum cardinality matching, that is, a matching \( M \) such that for any other matching \( M' \) we have \(|M| \geq (1 - \varepsilon)|M'|\). As the procedure keeps the maximum rank \( R \) bounded by \( O(\varepsilon^{-1}) \), it works in \( O(|E| \cdot \varepsilon^{-1}) \) total time and generates paths of \( O(|V| \cdot \varepsilon^{-1}) \) total length.

It is worth pointing out, that both \( O(|E| \cdot |V|^{1/2}) \) and \( O(|E| \cdot \varepsilon^{-1}) \) happen to be the same as the asymptotic times needed by the well-known algorithm of Hopcroft and Karp [13] to compute the exact and \( (1 - \varepsilon) \)-approximate maximum cardinality matchings in offline setting. Thus, in Section 4.3 we compare the behavior of our algorithm to other offline matching procedures and formulate a conjecture that Algorithm 8, a randomized version solving the offline maximum bipartite matching problem, has expected running time of \( O(|E| \cdot |V|^{\alpha}) \) time for some \( \alpha < 1/2 \).

**Conjecture 4.7.** If the vertices arrive in a random order, then the ranks and tiers technique yields an algorithm that runs in \( O(|E| \cdot |V|^{\alpha}) \) time for some \( \alpha \) strictly smaller than \( 1/2 \).

We finish Chapter 4 with two reductions: from decremental-only case and from weighted case. The first considers a setting in which we are given a bipartite graph
G and each turn instead of adding a vertex we remove one. In Section 4.4 we show that Algorithm 3 and Algorithm 4 can be transformed to handle the decremental case in the same running times. Then, in Section 4.5 we use the unfolded graph technique of Kao, Lam, Sung and Ting [14] to apply all the above algorithms and their modifications to weighted graphs. In this way we obtain an incremental and decremental algorithms that maintain the exact weights of the maximum weight bipartite matchings in $O\left(W^{3/2} \cdot |E| \cdot |V|^{1/2}\right)$ and their $(1 - \varepsilon)$-approximations in $O\left(W \cdot |E| \cdot \varepsilon^{-1}\right)$ total time (the letter $W$ denotes the maximum weight of an edge).

Finally, Chapter 5 presents a number of examples that demonstrate various characteristic behaviors of Algorithms 3 and 4, and provide lower bounds on the pessimistic ranks and tiers. We consider two methods of causing high ranks, one that targets the searching procedure, and another that focuses on the augmenting paths.

The former strategy is discussed in Section 5.1, in which we construct for a given parameter $r$ three problem instances that achieve a cubic sum of ranks $\Omega(r^3)$ while being of sizes $O(r^2)$, $O(r^2)$ and $O(r^2 \log r)$ respectively. We start with an exposition of a general idea using a simple example for a version of our matching algorithm that is based on the breadth-first search. Then we proceed to modify it to accommodate Algorithm 3, which is based on the depth-first search. However, the second example works only in the worst-case. To this end we introduce the third example, that achieves an expected $\Omega(r^3)$ sum of ranks even if the edges of any vertex are examined in a random order.

The shortcoming of these examples is that, the total length of applied augmenting paths is linear in $|V|$. Thus, in Section 5.2 we construct a problem instance of size $O(r^2)$ that causes both the total sum of ranks and the total length of augmenting paths to be $\Omega(r^3)$. As the construction is quite complicated, we first characterize its basic building blocks, and then demonstrate how to use them to achieve ranks linear in $r$ with an example of size $O(r^3)$. We end Chapter 5 with a description of how to compress the last instance to achieve the same effect within $O(r^2)$ size, which proves that the worst-case analysis of Algorithm 3 is tight.

References


