University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Computer Science

Analysis of Deterministic Algorithms for Online Metric Matching

Supervisor
Prof. Michele Scquizzato

Co-supervisor
Ing. Enoch Peserico

Candidate
Alberto Nicoletti

Academic Year
2023-2024

To my teachers
and my students

Abstract

Online Metric Matching is an online problem in which there are k servers and n requests placed on a metric space. The positions of the requests are revealed one by one and an online algorithm has to irrevocably match one request to an available server as soon as it's revealed. The goal of the algorithm is to minimize the total cost of the matching it makes, where the cost of a single match is the distance between the request and the server. An online algorithm is evaluated by computing the ratio of the total cost of its matching over to the total cost of the ideal optimal offline solution. Such ratio is called competitive ratio and an online algorithm is defined as c-competitive if it can achieve a competitive ratio of c. This thesis provides a re-parametrized analysis of the two most studied online algorithms for Online Metric Matching: Nearest Neighbor and Permutation. We provide demonstrations for known but never proved results. In particular, we prove that Permutation can't be $O(k)$-competitive by providing an instance where it is $\Omega(n)$-competitive for k as low as four. Additionally, we show that Nearest Neighbor is $(2^k - 1)$-competitive.

Contents

Listing of figures

1 Introduction

Online Metric Matching is a classic problem in the context of online algorithms. We have a bipartite graph in which nodes represent n requests and k servers of different capacities placed on a metric space. The positions of the requests are unknown at the beginning and they are revealed one by one in an online manner. The problem requires to irrevocably match each request to an available server as soon as the position of the request is revealed. The goal is to minimize the total cost of the matchings, where the cost of a single match is the distance between the request and the server. This problem can also be considered as the problem of having n cars in a city and k different garages, each one with a known number of parking spots. The positions of the cars are revealed one at a time and every car needs to be immediately assigned to an available parking spot.

The problem was simultaneously introduced by Kalyanasundaram et al. [1] and Khuller et al. [2] in 1993 and it still is constantly studied to this day, with different algorithms and different versions of the problem. The first definition of the problem considered n requests and n servers but in 1996 Kalyanasundaram and Pruhs [3] proposed a version of the problem in which we have k servers with a capacity of b and n requests, with $n = kb$. In this thesis we consider a more general context with k servers of possibly different capacities and n requests. Note that an analogous formulation of the problem is to consider n requests and n servers placed on k distinct server locations, with $k < n$.

The two most studied algorithms for Online Metric Matching are the natural greedy algorithm Nearest Neighbor and the (almost) optimal algorithm Permutation. The performance

of an online algorithm is measured by the competitive ratio, which is the ratio between the final cost of the algorithm and the ideal optimal solution. Originally, the competitive ratios of Nearest Neighbor and Permutation were studied as functions of n. The focus of this thesis is to do a re-parametrized analysis of the competitive ratio of the two algorithms. We will see that the competitive ratio of Nearest Neighbor can be re-parametrized as a function of k, while such re-parametrization is not possible for Permutation.

In Chapter 2 we introduce the concept of online problems and we define the problem of Online Metric Matching. In Chapter 3 we provide an overview of the state of the art of the problem. We summarize our contribution in Chapter 4. Chapter 5 provides some basic background for the following demonstrations. In Chapter 6 we do a re-parametrized analysis of Nearest Neighbor and in Chapter 7 we analyze Permutation. We conclude with Chapter 8, providing some final considerations and a future research agenda.

2

Definition of the Problem

This chapter is dedicated to provide definitions and notations that are the basis for the thesis. We begin recalling the concept of optimization problems and their online version. Then we describe the competitive analysis of online algorithms and the concept of competitive ratio. Finally, we define the problem investigated in this thesis: Online Metric Matching.

2.1 Online Algorithms and Competitive Analysis

An optimization problem can either be of cost minimization or profit maximization, in this thesis we focus on a cost minimization problem. An optimization problem $\mathcal{P}$ of cost minimization consists in a set $\mathcal{I}$ of inputs and a cost function C. Each input $I \in \mathcal{I}$ is associated with a set of feasible solutions (or outputs) $F(I)$ and each feasible solution $O \in F(I)$ is associated with a positive real number $C(I, O)$, that is the cost of the output O with respect to the input I.

Given an input I, the goal of an algorithm ALG for a minimization problem is to compute a feasible solution $ALG[I] \in F(I)$ that minimizes the cost $ALG(I) = C(I, ALG[I])$. An optimal algorithm, denoted as OPT, always provides a solution that minimizes the cost, that is, for all possible inputs,

$$OPT(I) = \min_{O \in F(I)} C(I, O).$$

Online problems are optimization problems in which the input is received in an online man-

ner and the output must be produced online as well. In an online problem I is in fact defined as input sequence. An algorithm ALG for an online problem is called online algorithm. An online algorithm does not have all the input sequence I available since the beginning of its computation, but it has only partial knowledge of it. At each time step t of the computation, a previously-unknown part of I is revealed to ALG and the algorithm must make an irreversible decision, that will have an impact on the final performance of the algorithm.

Optimization problems can be partitioned in online problems and offline problems and both classes have many relevant real-world scenarios. Many instances of linear programming are usually offline, while problems like paging are intrinsically online. There is then a big class of problems, like job scheduling or bin packing, that are meaningful both in their offline and their online version.

In this thesis we focus only on deterministic algorithms. To understand the performances of a deterministic online algorithm we introduce the concepts of competitive ratio and competitiveness. The best way to measure the performance of an online algorithm ALG is to compare the cost of its solution with the cost of the solution of an hypothetical optimal offline algorithm OPT with clairvoyant abilities.

An online algorithm ALG is c-competitive if there is a constant α such that for all finite input sequences I,

$$ALG(I) \leq c \cdot OPT(I) + \alpha$$

We can also say that ALG has a competitive ratio of c. This means that, even in the worst-case scenario, the cost of the solution produced by the online algorithm is at most c times (plus a constant α) the cost of the optimal solution. In the context of the analysis of online algorithms it is useful to see the online problems as games between an online player (ALG) and an adversary that creates an input sequence I that makes ALG incur high costs while keeping the costs for OPT low.

2.2 Online Metric Matching

To introduce the online problem that is the focus of the thesis, we give a definition of metric space.

Definition 2.2.1 (Metric Space)**.** A metric space is an ordered pair (M, d) where M is a set of

points and d is a metric on M, i.e., a function

$$d : M \times M \to \mathbb{R}$$

satisfying the following properties for all points $x, y, z \in M$

1. $d(x, x) = 0$
2. If $x \neq y$, then $d(x, y) > 0$ (positivity)
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequality)

In the Online Metric Matching problem - also known as Weighted Matching, Metric Bipartite Matching, Facility Assignment or Transportation Problem - there is a complete bipartite graph G in which the nodes of one partition are called servers and the nodes of the other partition are called requests. The nodes represent points in a metric space and the edges' weights are the distances between those points. There are k servers and n requests, with $k \leq n$ and each server s_i has a capacity c_i, for $i = 1 \ldots k$. Each server s_i can be matched to c_i requests and $\sum_{i=1}^{k} c_i = n$. The servers' positions are revealed immediately, while the requests are revealed one by one in an online manner. As soon as a new request is revealed the online algorithm ALG has to irrevocably match it with an available server. A server s_i is available if it is matched with less than c_i requests.

In this problem the cost function is defined to be the sum of the weights of the edges selected as matches by the algorithm. Given an input sequence I, $ALG(I)$ is the sum of the distance between each request and its matched server in the online algorithm and $OPT(I)$ is the sum of the distances between each requests and its matched server in the optimal solution. Where no ambiguity will result, we will drop the reference to the input sequence and refer to the total online and offline costs as ALG and OPT.

When all servers have an equal capacity of b we call the problem online metric b-matching. The Online Metric Matching problem was simultaneously introduced by Khuller et al.[2] and by Kalyanasundaram and Pruhs [1] in a version where each server has unit capacity. This problem is not different from our problem, in fact considering k servers with different capacities is the same as considering n servers of unit capacity placed in k different locations. The focus of the thesis is the re-parametrization from n to k.

Like most matching problems, Online Metric Matching is considerably important as it can have numerous applications. In Chapter 1 we already explained how this problem can be applied to cars looking for a parking spot. Other applications are: dispatching ambulances and fire trucks to emergencies response, handling user requests in a bike-sharing app, managing order picking in warehouses and coordinating urban trash collection.

3
Related Work

Matching problems are among the most studied problems in computer science. Considering matching in an online version in a metric space is of high interest, and as a result, extensive literature is devoted to this problem. In this chapter we summarize some of the most relevant results in the field to understand the state of the art of the problem. This thesis mostly focuses on the general problem. Nevertheless, in Section 3.1 we will also briefly present interesting results in other variants of Online Metric Matching.

Kalyanasundaram and Pruhs [1] presented the problem assuming servers of unit capacity. In the same article, the authors provided the definitions of the two most studied algorithms for this problem: Nearest Neighbor (often referred to as greedy) and Permutation. They proved that Nearest Neighbor is $(2^n - 1)$-competitive providing matching upper and lower bounds. In the context of online maximum matching Nearest Neighbor was shown to have an optimal competitive ratio of 3. Permutation was proved to be $(2n - 1)$-competitive and the authors also provided a general lower bound of $2n - 1$ on the competitive ratio of any deterministic algorithm, showing that Permutation is optimal. This work is of fundamental importance for this problem and some of its intuition will come back in this thesis, in different forms. Khuller et al.[2] independently worked on the same problem and also presented the online algorithm Permutation.

In a chapter of a book from 1996, Kalyasundaram and Pruhs [3] presented the b-matching problem and stated, without proof, that Nearest Neighbor is still $(2^k - 1)$-competitive in this version of the problem, while the competitive ratio of Permutation rises to $\Theta(kb) = \Theta(n)$.

The main contribution of this thesis is providing proofs for these statements. The authors posed an open question about the possible existence of an algorithm that is $O(k)$-competitive and conjectured that there should be a $(2k-1)$-competitive algorithm. This long-unanswered question found a nearly optimal solution in 2024. Harada and Itoh [4] presented the algorithm Subtree-Decomposition, that achieves a competitive ratio of $8k-5$ in the general setting of Online Metric Matching with servers of different capacities. In this thesis we provide a lower bound of $2k-1$ for any deterministic algorithm which means that there still is an open gap between our $2k-1$ lower bound and the $8k-5$ upper bound of Subtree-Decomposition.

One of the most interesting special cases of this problem is the Online Metric Matching on a line, which considers the real line as metric space. Nearest Neighbor maintains the exponential lower bound of 2^k-1 but if we consider the equal line as the metric space, i.e. the distance between two adjacent servers is always one, Ahmed et al. [5] showed that the algorithm achieves a competitive ratio of $4k$ and Harada et al. [6] provided a $4k-5$ lower bound. Ahmed et al. [5] also claimed with rough proofs that Permutation is k competitive on the equal line but Harada [7] later provided a lower bound of $k+1$. This result disproves the upper bound of k, leaving open the question of whether an upper bound exists on the competitive ratio of Permutation on the equal line as a function of k.

For real line with arbitrary distances, Peserico and Scquizzato [8] provided a lower bound of $\Omega(\sqrt{\log n})$ on the competitive ratio of any algorithm (both randomized and deterministic) on the line with servers of unit capacities, which is the best known result so far. The upper bound for this problem is $\Theta(\log n)$ which is the competitive ratio of the deterministic algorithm Robust Matching Algorithm introduced by Nayyar and Raghvendra [9], with an improved analysis by Raghvendra [10]. Antoniadis et al. [11] showed that online matching on a line is essentially equivalent to a particular search problem called k-lost cows and provided a deterministic sub-linearly competitive algorithm for online matching on a line by giving an algorithm for the k-lost cows problem. This algorithm is a reasonably natural greedy algorithm, but the resulting algorithm for Online Metric Matching is not particularly intuitive, supporting the hypothesis that it is easier to reason about online matching via search rather than online matching directly. Another interesting result for the line is the algorithm PTCP (Policy Transition at Critical Point) by Harada and Itoh [12]. The authors proved that PTCP is $(2\alpha(S)+1)$-competitive, where $\alpha(S)$ is informally the ratio of the diameter of the metric space to the maximum distance between two adjacent servers. Depending on the layout of servers, the competitive ratio ranges from constant to $2k-1$. Harada et al. [6] showed that there exists a class of algorithms that naturally generalizes the greedy algorithm called MPFS (Most

Preferred Free Server) and showed that any algorithm of this class has the capacity-insensitive property. This property means that an MPFS algorithm is c-competitive when servers have unit capacity if and only if it is c-competitive when each server has a capacity of b. Note that Subtree-Decomposition [4] is a MPFS algorithm.

3.1 Further Related Work

Unweighted online matching is a well-known problem, apparently similar to Online Metric Matching but it is significantly different both in theory and in its applications. In this problem we are trying to maximize the number of matches in an unweighted bipartite graph, where the nodes of one partition and their relative edges are revealed in an online manner. For this problem, Kalyanasundaram and Pruhs [13] presented the optimal deterministic algorithm Balance, which achieves the optimal competitive ratio of $1 - 1/(1 + 1/b)^a$ where a is the capacity of online servers and b is the capacity of adversarial servers. Kalyanasundaram and Pruhs [14] also showed that Balance is $O(1)$-competitive in the problem of Online Metric Matching under the assumption that each server's capacity is halved for the adversary but not for the online algorithm. Under the same assumption Nearest Neighbor is shown to be $\Theta(min(k, \log n))$-competitive.

In the problem of Online Bottleneck Matching, like in the Online Metric Matching, there are k servers in a metric space and k requests that are revealed in an online manner and needs to be matched to an available server. The objective is to minimize the maximum distance between any request and its matched server. Anthony and Chung [15] studied the performances of Nearest Neighbor, Permutation and Balance in this context, both against a normal adversary and against a weaker adversary under the assumption that each server has a capacity of two but only for the online algorithm. The authors proved that the competitive ratio of Nearest Neighbor improves from exponential to linear against a weaker adversary, while the competitive ratios of Permutation and Balance remains linear.

Recourse is a small modification that can be applied to achieve surprisingly good results. In Online Metric Matching with Recourse, the online algorithm is allowed to change a small number of previous matches upon the arrival of a new request. Considering servers of unit capacity, Gupta et al. [16] showed that allowing a logarithmic ($O(\log k)$) amount of recourse can bring to a $O(\log k)$-competitive algorithm in general metrics and a 3-competitive algorithm in the line metric. Interestingly, these results are achieved with small modifications on Permu-

tation. Megow et al. [17] provided an $O(1)$-competitive algorithm for online matching on the line with amortized recourse of $O(\log n)$ and a $(1+\varepsilon)$-competitive algorithm that reassigns any request at most $O(\varepsilon^{-1.001})$ times under the assumption that there are no more than one request between two servers.

Finally we discuss the best results for Randomized Metric Matching. Considering an oblivious adversary that has to choose the input sequence in advance playing against a randomized algorithm can lead to significantly stronger algorithms. The best known lower bound for the competitive ratio of a randomized algorithm is $\Omega(\log k)$. Bansal et al. [18] provided a $O(\log^2 n)$-competitive randomized algorithm when each server has unit capacity and Kalyanasundaram et al. [19] showed a $O(\log^2 k)$-competitive randomized algorithm when each server has a capacity of b.

4

Our Contribution

In this chapter we list the results and contributions of our work. First we provide a general lower bound of $2k - 1$ for any deterministic algorithm. A bound of $2n - 1$ is known since 1993 [1], while our bound for the general problem is new.

Next, we analyze the algorithm Nearest Neighbor showing that it is $(2^k - 1)$-competitive. Kalyanasundaram and Pruhs [3] already stated this result but no proof was ever published. We provide complete proofs for the lower bound and upper bound showing that the bound is tight. The upper bound is of particular interest because it shows that the original competitive ratio of $2^n - 1$ can be re-parametrized as a function of k, while a similar re-parametrization is not possible for Permutation.

The upper bound of $2^k - 1$ for Nearest Neighbor is proved with an inductive proof, using a similar technique to the one used by Kalyanasundaram and Pruhs in their original paper [1]. The authors use an inductive proof to also provide the $2n-1$ upper bound for Permutation. In this thesis we attempt an inductive proof for a possible $2k-1$ upper bound for Permutation and show why this proof is wrong and which is the point that makes such proof not possible. This highlights a valuable difference in the two most important and studied algorithms for Online Metric Matching.

A common intuition that could lead to an upper bound of Permutation as a function of k is to divide the servers and the requests into disjoint and parallel metric spaces. We formally define this intuition as a conjecture and show that is wrong, failing even in a simple metric space such as the real line with equal distances.

The major contribution of this thesis is a lower bound for Permutation, showing that Permutation cannot be $O(k)$-competitive. This result is known since 1996 [3] but no proof was ever published. We provide an instance in which Permutation has a competitive ratio of $\Omega(n)$. This instance is set in the real line and this highlight the fact that Permutation could perform poorly even on a simple metric space like the line with arbitrary distances. This bound works for (almost) any choice of n and k and also for the problem of b-matching where all servers have the same capacity.

5 Preliminaries

In this chapter we provide some basic background for the rest of the thesis. First, we provide two definitions that will be useful in different proofs of the following chapters. Then, we show a general lower bound of $2k - 1$ for any deterministic algorithm using a star metric.

5.1 Exclusive-or, Alternating Paths and Cycles

Definition 5.1.1 (Exclusive-or, alternating path)**.** The *exclusive-or*, denoted $E_1 \oplus E_2$, of two subsets E_1 and E_2 of the edge set of a graph, contains those edges that are exactly one in E_1 and E_2. An *alternating path* (respectively, *cycle*) in $E_1 \oplus E_2$ is a simple path (respectively, cycle) with edges alternately in E_1 and E_2.

When servers have a capacity greater than one it is possible that there can be many ways of telling apart one alternating path/cycle from another. The correct way is to identify the requests that interact with each other during the run of the algorithms.

The concept of alternating paths or cycles will be used to understand the correlations between the matches made by an online algorithm and the matches of the optimal offline solution. In this direction, we add to the definition by providing a simple example of an alternating cycle in an instance for the greedy online algorithm Nearest Neighbor, which matches each request with the closest available server.

Consider three servers of unit capacity placed on the real line. Call the servers s_1, s_2 and s_3. The server s_1 is at position zero, s_2 is at position two and s_3 is at position four. The first request, r_1 arrives at position $1 - \varepsilon$ and is matched by Nearest Neighbor with s_1. The second request arrives at position zero and is matched with s_2. The last request arrives at position four and is matched to the last remaining server s_3. The optimal offline solution matches r_1 with s_2, r_2 with s_1 and r_3 with s_3. Having that the match (r_3, s_3) is made by both Nearest Neighbor and OPT, the exclusive-or $OPT \oplus NearestNeighbor$ contains one alternating cycle formed by the four edges (r_1, s_2), (r_2, s_2), (r_2, s_1) and (r_1, s_1). Figure 5.1 shows the configurations of the two algorithms.

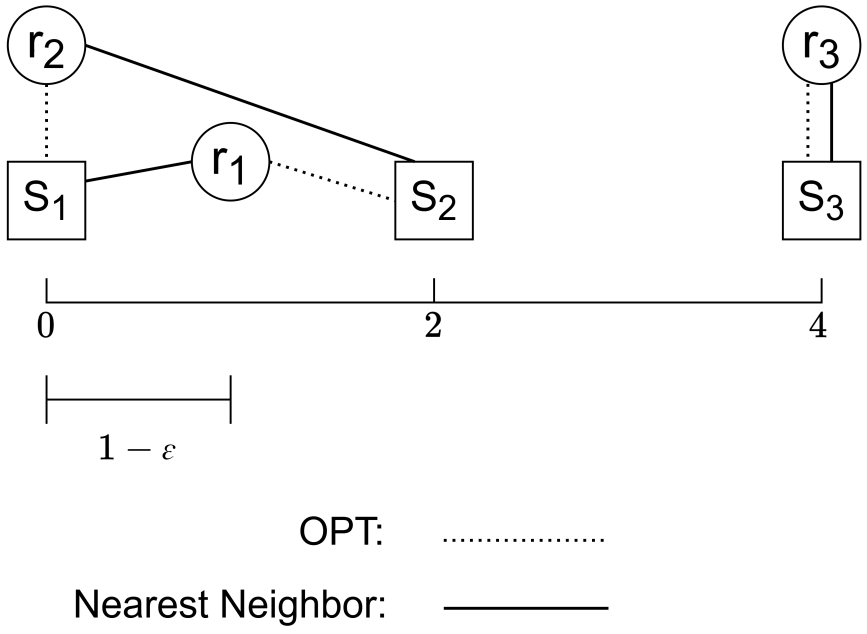

Figure 5.1: An example of configuation of OPT and Nearest Neighbor over a simple instance.

5.2 Partial Matching

We define the concepts of partial matching and minimal weight partial matching as follows.

Definition 5.2.1 (partial matching)**.** Given a subset R_i of the first i requests and the set of servers S, the *partial matching* of R_i is a matching of all requests in R_i with a subset of S. The *minimal weight partial matching* of R_i is denoted as M_i and is the partial matching of R_i with minimal weight and with a minimal number of edges in $M_i - M_{i-1}$. We define M_0 to be empty.

Note that M_i is the partial matching that OPT makes after i requests are revealed.

5.3 A Lower Bound for any Deterministic Online Algorithm

Theorem 1. *For every k, number of servers, there exists an instance for the problem of Online Metric Matching with the property that the competitive ratio of any deterministic online algorithm is at least $2k - 1$.*

Proof. The following construction is similar to the one provided by Kalyanasundaram and Pruhs [1] when showing a bound of $2n - 1$. We consider as metric space a star metric with one root and k leaves, a distance of one between the root and any leaf and a distance of two between any two leaves, similar to the example in Figure 5.2.

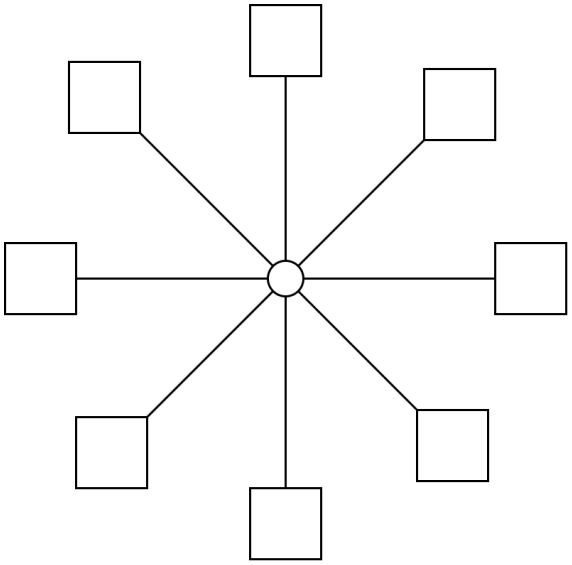

Figure 5.2: An example of start metric with 8 servers on the leaves and one request on the root.

Let ALG be any deterministic online algorithm for Online Metric Matching and OPT be the optimal offline algorithm. We will consider the behavior of an adversary deciding the positions of the requests. There is one server at each leaf. The first $n - k$ requests appear on the leaves. On top of any server s_i, with capacity c_i, place $c_i - 1$ requests. From here we distinguish two possible cases.

First case: ALG matches each request with the server on the same leaf with a total cost of zero, leaving every server with a remaining capacity of one. The next request arrives on the root

node and ALG matches it with one server on a leaf with a cost of one. One server is now fully occupied. The adversary from now on will place a request on the most recently occupied server, so that ALG must match the request with a server of another leaf paying a cost of two. The overall cost of ALG is

$$1 + 2(k-1) = 2k - 1.$$

Second case: ALG does not match one or more of the first $n - k$ first requests with the server on the same leaf. This incurs a cost greater than zero. There are now some servers fully occupied and some with a remaining capacity of at least one. The behavior of the adversary is the following:

If there is a leaf with a fully occupied server s_i where $c_i - 1$ requests have been placed so far, place a request on this leaf. Otherwise place one request on the root.

In this case ALG will match, with a cost of two, each request from a leaf with a fully occupied server with the server on another leaf. Then, we reach a point in which there are some leaves with c_i requests and a fully occupied server s_i and the remaining leaves where $c_{i'} - 1$ requests have been placed and a server $s_{i'}$ with a remaining capacity of one. Now the adversary places a request on the root and proceeds as in the *first case*. The total cost of ALG is at least $2k - 1$.

OPT matches all requests on a leaf with the server on the same leaf and the one request on the root with the server s_i on the leaf where $c_i - 1$ requests are placed, with a total cost of one. Thus, we showed that any deterministic online algorithm for Online Metric Matching has a competitive ratio of at least $2k - 1$. □

It is interesting to show that the same metric space can be used to find a lower bound in function of k using another technique. In 2017, Nayyar and Raghvendra [9] showed that a lower bound for any online algorithm can be found by having a good set of servers in a metric space, with the following theorem.

Theorem 2. *Given a set of servers S in a metric space M, any online algorithm ALG will have a competitive ratio of $\Omega(\mu_M(S))$.*

Given a set of point W, $DIAM(W)$ is the maximum distance between any pair of points in W and $TSP(W)$ is the smallest cost simple cycle that visits every point in W. Given the set of servers locations S, $\mu_M(S)$ is defined to be

$$\mu_M(S) = \max_{W \subseteq S, DIAM(W) > 0} \frac{TSP(W)}{DIAM(W)}$$

Given this definition is easy to see that the star metric, with its low diameter, can be used to find a good lower bound. In particular, the star metric we previously used has $DIAM(S) = 2$ and $TSP(S) = 2k$, providing a lower bound of k.

Nayyar and Raghvendra [9] proved Theorem 2 for the case where each server has a capacity of one. We now prove that Theorem 2 is also true when servers can have different capacities. We first use the simple technique of filling all servers until each has a remaining capacity of one and then we can continue with the same proof as the one in [9].

Proof. Let $S' \subseteq S$ such that $\mu_M(S') = \mu_M(S)$. We can remove any server $s_{i'}$ that is not in S' by placing $c_{i'}$ request on that server. Any match for ALG that is not with $s_{i'}$ will result in an higher cost for the algorithm. We then place, at the position of each server s_i, $c_i - 1$ requests and we can assume that they will be matched by ALG with s_i because any other match will also results in an higher cost for the algorithm. The next two requests r_1 and r_2 are placed at the position of some server s_1. ALG will match the r_1 with s_1 with a cost of zero and r_2 with a server s_2. Next we can continue placing a request r_i at the position of the lastly matched server s_{i-1}, for $i = 3..k$. The total cost incurred by ALG is $\sum_{i=1}^{|S'|-1} dist(s_i, s_{i+1})$ which is the cost of a path that visits every vertex in S'. Therefore, ALG pays at least $TSP(S')/2$. OPT can instead match each r_i with s_{i-1} for a cost of zero and will match r_1 with $s_{|S'|}$ with a cost that is at most $DIAM(S')$. □

6
Nearest Neighbor

In this chapter we provide a re-parametrized analysis for the competitive ratio of the greedy algorithm Nearest Neighbor. We first describe the algorithm and then we provide matching lower and upper bound to prove that Nearest Neighbor is $(2^k - 1)$-competitive.

6.1 The Algorithm

Nearest Neighbor is an online algorithm for metric matching. Every new request is matched by the algorithm with its closest available server, breaking ties arbitrarily. More precisely, every new request r_i is matched with the server $s_{i'}$, such that

$$s_{i'} = \arg\min_{s \in S_{free}} d(r_i, s)$$

where S_{free} is the set of servers that have a capacity greater than the number of requests that are currently matched to it, i.e. those servers that are not completely occupied.

We now show that Nearest Neighbor is $(2^k - 1)$-competitive and that this bound is tight. Kalyanasundaram and Pruhs [1] proved this result for the special case in which the capacity of each server is one. We provide an upper bound in Theorem 3 and a lower bound in Theorem 4 for the general problem of Online Metric Matching.

6.2 The Upper Bound

Theorem 3. *Nearest Neighbor achieves a competitive factor of* $2^k - 1$.

Proof. Recalling definition 5.2.1, let N_i be the partial matching constructed by Nearest Neighbor after the request that makes the i-th server completely occupied. Let M_i be the partial optimal matching of the requests in N_i. We define N_0 and M_0 to be empty. Where no ambiguity will results, we also use M_i and N_i to denote the weight of the relative matchings. We prove inductively that $N_i \leq (2^i - 1)M_i$. For $i = 1$, $N_i = M_i$ and the result follows. Now we assume that the result holds for $i - 1$ and we verify it holds for i. We define R_i to be the set of requests of the partial matching $N_i - N_{i-1}$. Nearest Neighbor matches each request $r_\ell \in R_i$ with a server noted as s_ℓ. Call s_{OPT_ℓ} the server matched to r_ℓ in M_i. Note that these servers are not necessarily distinct.
We distinguish two cases. If $N_i - N_{i-1} \leq M_i - M_{i-1}$ then:

$$\begin{aligned}
N_i &\leq N_{i-1} + M_i - M_{i-1} \\
&\leq (2^{i-1} - 1)M_{i-1} + M_i - M_{i-1} && \text{(by inductive hypothesis)} \\
&\leq (2^{i-1} - 2)M_{i-1} + M_i \\
&\leq (2^{i-1} - 1)M_i && \text{(using } M_{i-1} \leq M_i\text{)} \\
&\leq (2^i - 1)M_i
\end{aligned}$$

In the second case, we consider if $N_i - N_{i-1} > M_i - M_{i-1}$. Each r_ℓ requests of R_i is either matched with the same server both in M_i and in N_i or not. Consider any request r_ℓ matched with the same server in N_i and in M_i. We have that $d(r_\ell, s_\ell) = d(r_\ell, s_{OPT_\ell})$ since the two servers are actually the same server. Note that (r_ℓ, s_{OPT_ℓ}) is a unique edge of M_i. Now consider any request r_ℓ that is matched differently in N_i with respect to M_i. We have that s_{OPT_ℓ} is incident to an edge in N_{i-1}, since it was not available for Nearest Neighbor at the arrival of r_ℓ. Let H be the exclusive-or of N_{i-1} and M_i. There is a unique alternating path that goes from r_ℓ to a server s_{f_ℓ} that is matched in M_i and not full in N_{i-1}. If the server s_{f_ℓ} coincides with s_ℓ we have that the sum of the weights of the edges in the alternating path in H is greater or equal than $d(r_\ell, s_\ell)$ by the triangular inequality. If the server s_{f_ℓ} does not coincide with s_ℓ we have that $d(r_\ell, s_\ell) \leq d(r_\ell, s_{f_\ell})$ by definition of Nearest Neighbor and $d(r_\ell, s_{f_\ell})$ is less or equal than the sum of the weights of the edges in the alternating path in H by the triangular inequality. Combining all of the above, we have that the overall cost of the matches made by Nearest Neighbor of the requests in R_i is

not greater than the overall cost of the matches in H, which gives us that $N_i - N_{i-1} \leq N_{i-1} + M_i$ and we use it to show that:

$$\begin{aligned} N_i &\leq M_i + 2N_{i-1} \\ &\leq M_i + 2(2^{i-1} - 1)M_{i-1} && \text{(by inductive hypothesis)} \\ &\leq M_i + (2^i - 2)M_i && \text{(using } M_{i-1} \leq M_i\text{)} \\ &= (2^i - 1)M_i \end{aligned}$$

□

6.3 A Lower Bound

Theorem 4. *The competitive ratio of Nearest Neighbor is at least* $2^k - 1$.

Proof. We first recall the proof by Kalyanasundaram and Pruhs [1] for the special case in which we have n servers of capacity one. The metric space is the real line, with distances defined in the standard way. One server, s_1 is placed at -1 and each server s_i is placed at $2^{i-1} - 1$, for $2 \leq i \leq n$. The i-th request r_i is at position $2^{i-1} - 1$, for $1 \leq i \leq n$. Since Nearest Neighbor is deterministic, we can assume that it matches the first request r_1 with server s_2, breaking ties by choosing the rightmost server. The algorithm then matches each request r_i with the closest server on the right, s_{i+1}, and the last request r_n with s_1. The total cost of Nearest Neighbor is $2^n - 1$. The optimal solution matches r_i with s_i for $i = 1 \ldots n$. The total cost of the optimal matching is 1.

The proof for the case in which there are k servers with different capacity uses the same metric space and servers positions as the one provided in [1]. The first $n - k$ requests are placed at the same location of the servers. For $i = 1 \ldots k$ place $c_i - 1$ requests at the location of the server s_i. Nearest Neighbor matches each request with the server on the same position for a total cost of zero, leaving each server with a remaining capacity of one. The last k requests are defined as in the preceding paragraph, giving the desired competitive ratio of $2^k - 1$. Figure 6.1 shows the configurations of OPT and Nearest Neighbor after the arrival of the last k requests. The first $n - k$ requests are omitted for clarity. □

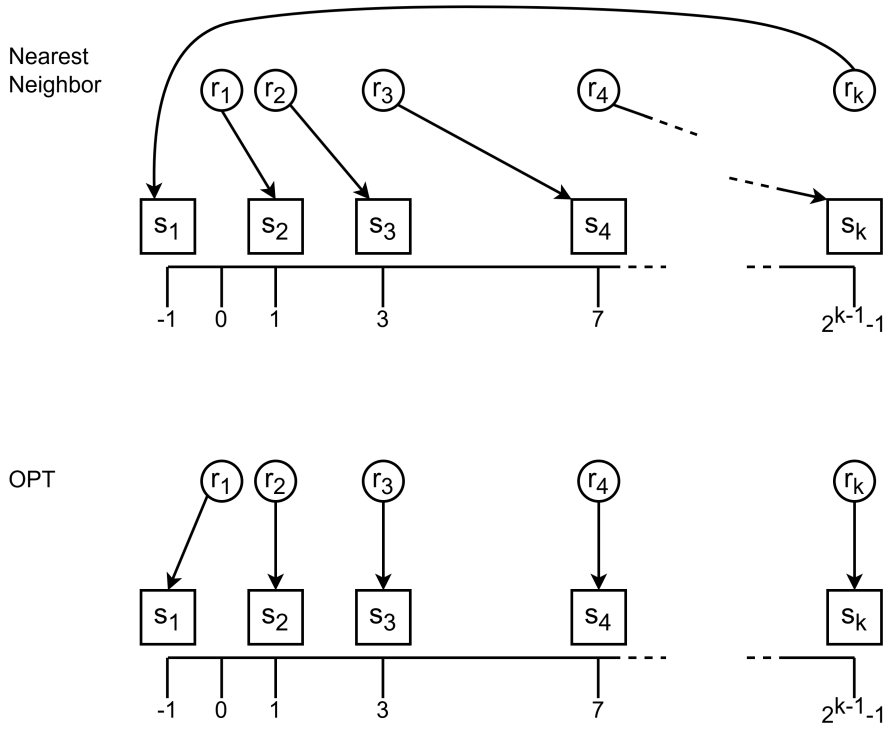

Figure 6.1: A Lower Bound instance for Nearest Neighbor.

7 Permutation

This chapter is dedicated to an analysis of Permutation in function of the parameter k. First we define Permutation, one of the most studied algorithms for Online Metric Matching. Then, we see two possible ways to provide a proof that could make Permutation $O(k)$-competitive and show why they are wrong. Finally, we give a $\Omega(n)$ lower bound showing that Permutation cannot be $O(k)$-competitive.

7.1 The Algorithm

Permutation was simultaneously defined by Kalyanasundaram et al. [1] and Khuller et al. [2]. To describe the algorithm is easier to consider all servers with unit capacity and k server locations. For example, a server with capacity four, acts in the same way as four servers of unit capacity placed at the same point of the metric space.

We recall from definition 5.2.1 that M_i is the partial optimal matching of the first i requests. Note that due to the offline nature of the optimal algorithm OPT, the matches it makes may change at the arrival of new requests. Let P_i be the partial matching of the first i requests constructed by Permutation. At the arrival of a new request Permutation checks whether it is on a location of a free server. If this is the case, Permutation matches the request with the server at the same position. Otherwise, Permutation responds to the $(i+1)$th request by computing

M_{i+1} and matching r_{i+1} with the unique server s that is in $M_{i+1} - M_i$. Permutation gets its name because it always maintains the following invariant:

For all i, the servers incident to an edge in P_i are exactly the servers incident to an edge in M_i.

7.2 Proof By Induction

We now provide an attempt of theorem that proves Permutation to be $(2k-1)$-competitive. This proof uses induction, like the one of Kalyanasundaram and Pruhs [1] to show the competitiveness of Permutation when capacities are equal to one. To achieve this result the authors used induction over the number of requests, while in the following proof the induction is over the number of servers that get closed by Permutation. In Chapter 6 we showed that an inductive proof is sufficient to prove an upper bound of $2^k - 1$ for the competitive ratio of Nearest Neighbor. Now we'll see that the same technique used for Permutation to show an upper bound of $2n - 1$, fails when attempting to prove an upper bound of $2k - 1$. At the end of the demonstration, we show the precise point in which the proof is wrong and provide a counterexample.

Attempt 1. *Permutation is $(2k-1)$-competitive.*

Proof. We prove inductively that the weight of P_i, the partial matching constructed by Permutation after the i servers are full, is at most $2i - 1$ times the weight of M_i, the partial matching of OPT of the requests that make i servers full in P_i. For $i = 1, M_i = P_i$ and the result follows.

Let's assume that the inductive hypothesis $P_{i-1} \leq (2(i-1) - 1)M_{i-1}$ is true. Let R_i be the set of the requests that arrives after P_{i-1} until P_i. Denote these requests as $r_{i_1}, r_{i_2}, \ldots, r_{i_t}$. Assume Permutation matches each r_{i_ℓ} with the server s_{j_ℓ} for $l = 1 \ldots t$. Note that the servers may not be all distinct. Let M' be the union of the partial matching M_{i-1} and all the pairs r_{i_ℓ} with the server s_{j_ℓ} for $l = 1 \ldots t$. Let H be the exclusive-or of M_i and M'. H contains t alternating cycles, one for each request in R_i, we denote their weights as $C_1, \ldots, C_t$. By the triangular inequality we have that, for every $l = 1 \ldots t, d(r_{i_\ell}, s_{j_\ell}) \leq C_\ell - d(r_{i_\ell}, s_{j_\ell})$. This gives

us that

$$\begin{aligned}\sum_{l=1}^{t} d(r_{i_\ell}, s_{j_\ell}) &\leq \sum_{l=1}^{t} C_\ell - d(r_{i_\ell}, s_{j_\ell}) \\ &= H - \sum_{l=1}^{t} d(r_{i_\ell}, s_{j_\ell}) \\ &\leq M_{i-1} + M_i\end{aligned}$$

Given that the sum of the matches of R_i is $P_i - P_{i-1}$, we have that $P_i - P_{i-1} \leq M_{i-1} + M_i$. Using this and the fact that $M_{i-1} \leq M_i$ we have

$$\begin{aligned}P_i &\leq M_{i-1} + M_i + P_{i-1} \\ &\leq M_{i-1} + M_i + (2(i-1)-1)M_{i-1} && \text{(by inductive hypothesis)} \\ &\leq 2M_i + (2(i-1)-1)M_i \\ &= (2i-1)M_i\end{aligned}$$

□

Now we show that the previous proof is wrong. In particular, there is one wrong statement that can be rephrased as:

Every cycle C_ℓ in H is an alternating cycle and it contains only one edge of the matches of the requests in R_i

from this statement we can then use the triangular inequality and reach to the desired result. To show that the statement is wrong we provide a counterexample in which there is an alternating cycle with two matches from $P_i - P_{i-1}$.

Consider the line as metric space and three servers, s_0, s_1, s_2, with equal capacity b, placed at equal distances at position $-1, 0, 1$. The first $b-1$ requests are at position zero and the next request, r_{0_b}, is at position $-\frac{1}{2}$. We can assume Permutation matches all these requests with s_1. We fully occupied one server and now we start with the requests of R_1. Let r_{1_0} be at position $\frac{1}{2} - \varepsilon$ for some $\varepsilon > 0$ and r_{1_1} be at position 0. Permutation matches r_{1_0} with s_0 and r_{1_1} with s_2. OPT matches the first $b-1$ requests with s_1, r_{0_b} with s_0, r_{1_0} with s_2 and r_{1_1} with s_1. Considering

H to be the exclusive-or of M_i and M' as in the proof above, there is an alternating cycle with two matches from $P_i - P_{i-1}$, precisely (r_{1_0}, s_0) and (r_{1_1}, s_2), as shown in Figure 7.1. Moreover, it is also clear to see that the inequality $P_i - P_{i-1} \leq M_i + M_{i-1}$ does not hold.

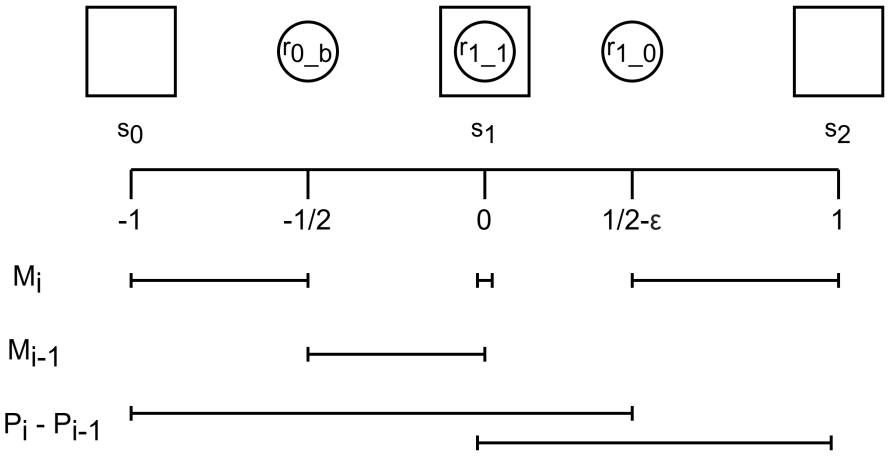

Figure 7.1: The alternating cycle in H with two edges from $P_i - P_{i-1}$.

7.3 A Conjecture

We now present a conjecture about Permutation. If proven true it would follow that Permutation is $O(k)$-competitive, even when servers have different capacities. We later provide a counterexample to the conjecture using the real line with equal distances as metric space. This shows that an idea that is apparently quite appealing as base for a demonstration cannot be used even in a simple metric space such as the equal line.

For simplicity of explanation, we only consider the special case of b-matching but the conjecture can be adapted also for the general case. Before providing the conjecture, we give an intuition that describes it. Consider the online problem of b-matching as the problem of parking n cars in k different garages and each garage can accommodate up to b cars. We can conjecture that this problem is equivalent to the problem of having a parking building divided in b floors with k parking places on each floor and each one of the n cars can arrive at any floor of our choice. The conjecture can be formally described as follows:

Conjecture 1. *Given any metric space and any placement of n requests and k servers of capacities b, it is possible to split the requests in b different instances of the same metric space with the same k*

servers but of unit capacities. There exists a split in which the matches of Permutation and OPT are the same as in the original metric space.

It's easy to see that if the conjecture is true it would prove that Permutation is $O(k)$-competitive, since any possible instance of Permutation could be split in different instances with servers of capacity of one, where Permutation is $(2k-1)$-competitive.

A counterexample to this conjecture can be done using the real line with equal distances. Let s_1, s_2, s_3 and s_4 be four servers placed on the line at an equal distance of one from each other. Place $b-1$ requests on top of s_2 and $b-1$ requests on top of s_3, leaving these two servers with a remaining capacity of one. The next request r_1 arrives in between of s_2 and s_3 and we can assume it is matched by Permutation with s_3. The next four requests are: r_2 at a distance of 0.5 to the right of s_3 and is matched with s_2; r_3 at a distance of 0.4 to the left of s_2 and is matched with s_4, r_4 at a distance of 0.3 to the right of s_3 and is matched with s_1 and finally r_5 at a distance of 0.2 to the left of s_2 and is matched with s_4. Figure 7.2 shows the arrival and matching of the last five requests.

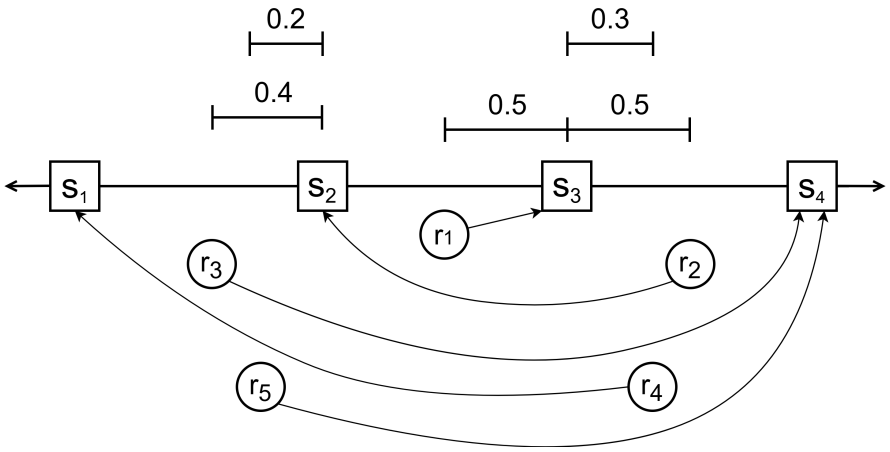

Figure 7.2: A counterexample for Conjecture 1.

Conjecture 1 does not work on this instance since it is not possible to split the requests in different instances in such a way that the behavior of Permutation and OPT remains the same. In particular, having that r_1 causes the expensive matchings, there is no split that would make both r_4 and r_5 be matched with s_4.

7.4 A Lower Bound for Permutation

We now prove that Permutation cannot be $O(k)$-competitive providing a lower bound that shows that Permutation is $O(n)$-competitive. We remark that the following Theorem is valid even in a simple metric space such as the real line and we can find an instance of the lower bound for any choice of n and any choice of k as low as 4.

The central idea behind the following construction is due to Kirk Pruhs [20]. The intuition of the proof is to make OPT change its matches at the arrival of every new request. This has the consequence of making Permutation do expensive matches. To achieve this, the first request arrives in the center and the following requests arrive on the left side and on the right side in an alternating way.

Theorem 5. *Permutation is $\Omega(n)$-competitive in the problem of Online Metric Matching.*

Proof. Consider the real line as metric space. Let $k \geq 4$, the number of servers be even and let n be the number of requests such that there exist a number $b \in \mathbb{N} \mid n = (k-2)b + 2$. This restrictions are useful to make an instance with two servers of unit capacity and the remaining servers with equal capacity of b symmetrically placed on the real line, which brings to an easier and more clear demonstration. The only necessary argument is that $k \geq 4$, removing all the others it is still possible to prove that Permutation is $\Omega(n)$-competitive.

The fist two servers, s_1 and s_2, are placed at positions $-n/2$ and $n/2$ respectively. Then, $(k-2)/2$ servers are placed to the left of s_1 with a distance of two between any two adjacent servers. Similarly, the remaining $(k-2)/2$ servers are placed to the right of s_2 with a distance of two between any two adjacent servers. All servers have a capacity of b but s_1 and s_2 have a capacity of one. We name each server on the left s_i, where i is an odd number and in increasing order from the center; and we name each server on the right $s_{i'}$, where i' is an even number and in increasing order from the center. Figure 7.3 shows the placement of requests for k=8.

For every $i = 1 \ldots n$ we denote the i-th request in order of arrival as r_i. The first request r_1 arrives at position zero. Then, r_2 arrives at a distance of $1 - \varepsilon_2$ to the right of s_2. For every $i = 2 \ldots n$, $\varepsilon_i > \sum_{j=2}^{i-1} \varepsilon_j > 0$ and $\varepsilon_n < 1$, making each request considerably closer to its nearest server than any previous request. Since each ε_i only depends on previous ε's, they all are infinitesimally small[1] and $(1 - \varepsilon_i) \approx 1$ for every $i = 2 \ldots n$.

The next request r_3 arrives at a distance of $1 - \varepsilon_3$ to the left of s_1 and r_4 arrives at a distance of $1 - \varepsilon_4$ to the right of s_2. Proceeding in a similar way, each even request r_i arrives at a distance

[1] For the sake of understanding the lower bound, the figures show epsilons of visible difference.

of $1 - \varepsilon_i$ to the right of s_2 and each odd request r_j arrives at a distance of $1 - \varepsilon_j$ to the left of s_1. This continues until b requests are placed in the line segment between s_3 and s_1 and b requests are placed in the line segment between s_2 and s_4. The procedure then continues placing alternatively one request to the left and one request to the right in such a way that between any two adjacent servers there are b requests and for any $j > i$, r_j is closer to a server than r_i. The last request is placed on top of the rightmost server. Generally, we can define that r_1 arrives at position 0, r_i is placed at position $(n/2) + 2 \cdot (\lceil i/2n \rceil - 1) + 1 - \varepsilon_i$ when i is even and at position $-(n/2) - 2 \cdot (\lceil i/2n \rceil - 1) - 1 + \varepsilon_i$ when i is odd for every $i = 2 \ldots n - 1$ and r_n is placed at $(n/2 + k - 2)$. Figure 7.3 shows the position of all requests for $k = 8$, $n = 20$ and $b = 3$.

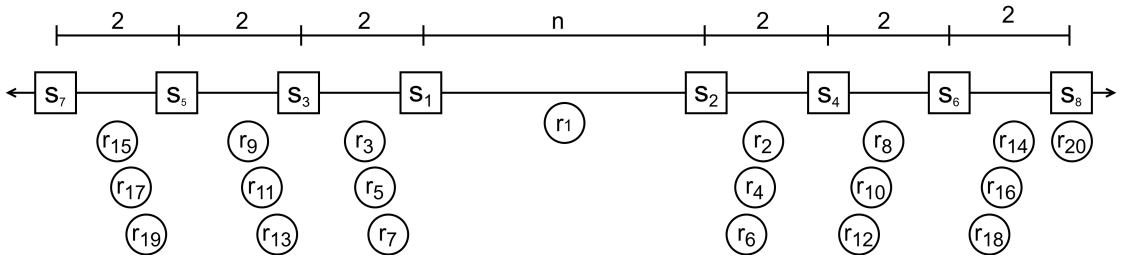

Figure 7.3: An example of a lower bound instance with $k = 8$ and $n = 20$.

We now describe the behavior of Permutation and OPT on the arrival of the first four requests. We can assume that Permutation matches the first request r_1 with s_2. We make this assumption because Permutation is deterministic. The proof for the case in which Permutation matches on the left when tie-breaking is symmetrical to the one we provide. Then, r_2 is matched to s_1 by Permutation because OPT matches r_1 with s_1 and r_2 with s_2. The next request r_3 arrives at a distance of $1-\varepsilon_3$ to the left of s_1 and is matched with s_4 by Permutation because the optimal offline configuration is matching r_1 with s_2, r_2 with s_4 and r_3 with s_1. At the arrival of r_4 OPT changes again its matches, forcing Permutation to match r_4 with s_3. Figure 7.4 shows step by step the behavior of OPT and Permutation on the arrival of the first four requests.

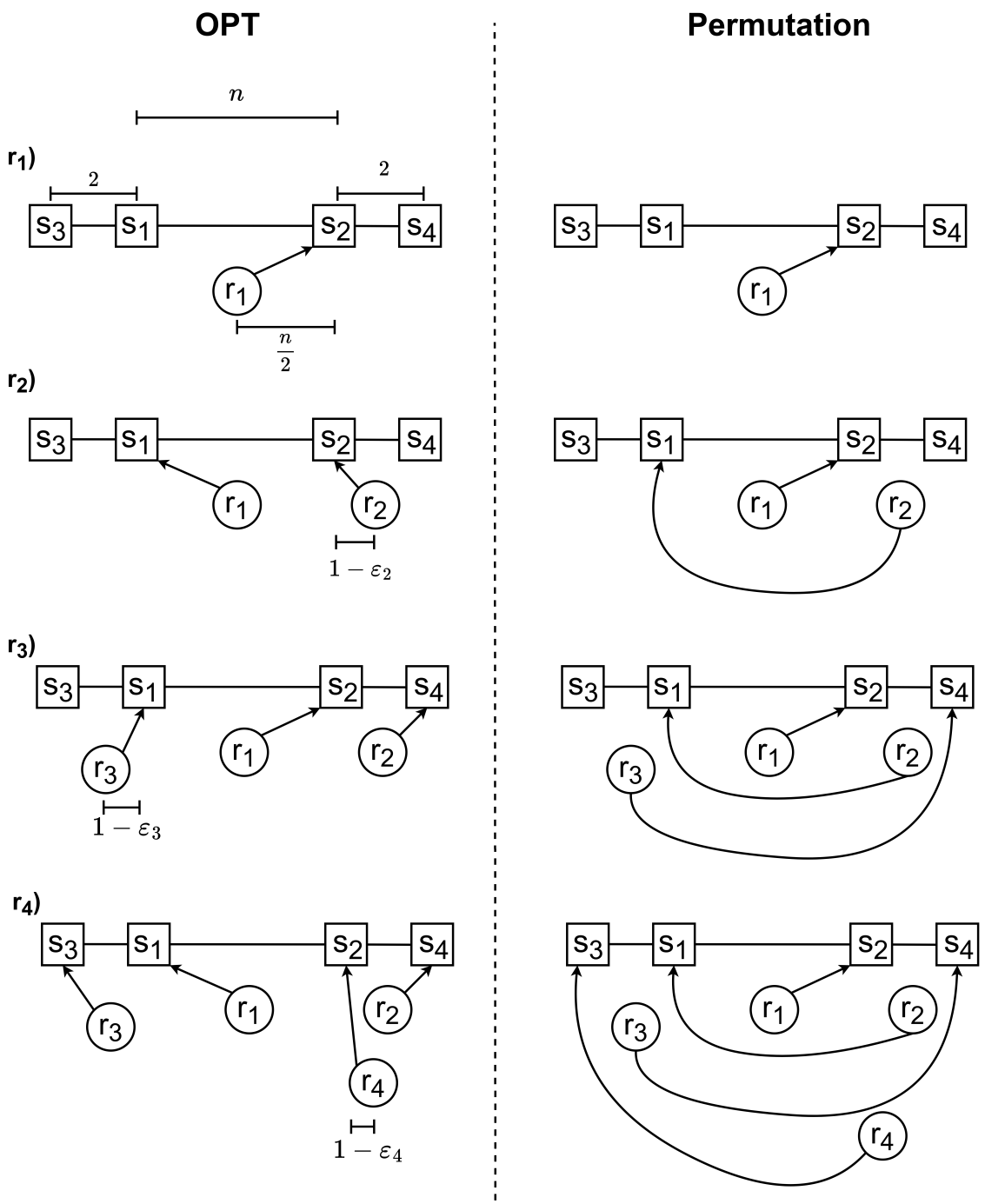

Figure 7.4: The configurations of OPT and Permutation for the first four requests.

OPT changes its matches at the arrival of every new request, which makes Permutation match every odd request placed on the left side of the line with an even server placed on the right side of line, and match every even request placed on the right side of the line with an odd server on the left side of the line.

We now formally prove that OPT flips -from right to left or from left to right- at the arrival of every request r_i, making Permutation match r_i with a server on the other side of the line. Consider that at the arrival of r_{i-1} OPT has matched r_1 with s_1, meaning that OPT is matching to the left. The next request r_i arrives on the left side of the line and we will see that OPT will flip causing r_i to be matched by Permutation with a server on the right side of the line. The proof of a flip of OPT from right to left is similar thus omitted. Call *section* the space between any two adjacent servers with distance two. The last request to arrive in a *section* is called *pivot*. At the arrival of r_i OPT has two options:

- Matching to the right: match r_1 with s_2, and match each *pivot* with the server on its right.
- Matching to the left: match r_1 with s_1, and match each *pivot* with the server on its left.

Considering that the matches of all requests that are not *pivot* do not change, and the cost for r_1 to be matched with s_1 or with s_2 is the same, the only difference in the costs of the two options is in the matches of the *pivots*. Call P_o and P_e the sets of indexes such that $j \in P_o$ if r_j is *pivot* and j is odd and $j' \in P_e$ if $r_{j'}$ is *pivot* and j' is even. Matching to the right adds a cost of

$$1 - \varepsilon_i + \sum_{j \in P_o} (1 - \varepsilon_j) + \sum_{j' \in P_e} (1 + \varepsilon_{j'})$$

and matching to the left adds a cost of

$$1 + \varepsilon_i + \sum_{j \in P_o} (1 + \varepsilon_j) + \sum_{j' \in P_e} (1 - \varepsilon_{j'})$$

Since $\varepsilon_i > \sum_{j=2}^{i-1} \varepsilon_j$, OPT will make a match on the right, causing it to flip and forcing Permutation to match r_i with a server on the other side of the line[2].

Denote *PERM* as the total cost of Permutation and *OPT* as the total cost of the offline algorithm. At the arrival of the last request, Permutation has matched r_1 with s_2 and each one

[2] From this proof we can also see that it's possible to define smaller epsilons making any ε_i greater than the sum of the epsilons of the *pivots* on the other side of the line minus the sum of the epsilons of the *pivots* on the same side of the line.

of the other $n-1$ requests with a server on the other side of the line for a cost that is at least n. This gives us that

$$PERM \geq \frac{n}{2} + n(n-1) = n(n - \frac{1}{2})$$

The optimal configuration of OPT is matching r_1 with either s_1 or s_2 for a cost of $n/2$ and each one of the remaining requests with an adjacent server at a cost that is not greater than two. The optimal offline cost is

$$OPT \leq \frac{n}{2} + 2(n-2) = \frac{5}{2}n$$

Thus, Permutation has a competitive ratio of at least

$$\frac{2}{5}(n - \frac{1}{2}) = \Omega(n).$$

□

8 Conclusion

In this thesis, we investigated the problem of Online Metric Matching. First, we provided a general lower bound for any deterministic online algorithm. Then, we focused on the two most studied online algorithms for this problem: Nearest Neighbor and Permutation. We provided a re-parametrized analysis for Nearest Neighbor, showing detailed proofs to the claim that Nearest Neighbor is $(2^k - 1)$-competitive with matching upper and lower bounds.

In the view of a re-parametrization of the competitive ratio of Permutation, we analyzed two possible demonstrations and showed how they fail in the attempt of proving a competitive ratio in function of k. Finally, we showed that Permutation can't be $O(k)$-competitive, providing a lower bound instance in which the competitive ratio of Permutation raises to $\Omega(n)$.

8.1 Further Research

We would like to highlight a few possible directions for future research on the problem. We proved that Permutation can't be $O(k)$-competitive, and Harada et al. [4] provided the algorithm Subtree-Decomposition that achieves a competitive ratio of $8k - 5$. In Section 5.3 we proved a general lower bound of $2k - 1$ for the competitive ratio of any deterministic online algorithm. An open question remains whether the gap between the competitive ratio of Subtree-Decomposition and the lower bound can be reduced, either through an improved analysis of the algorithm or by proposing a new online algorithm. When stating that Permutation can't be

$O(k)$-competitive, Kalyanasundaram and Pruhs [3] conjectured that there should be a $(2k-1)$-competitive algorithm.

An interesting research direction for a re-parametrization of the competitive ratio of Permutation is to consider the real line with equal distances as a metric space. In Chapter 3 we showed that Permutation was claimed to be k-competitive on the equal line, but a lower bound of $k+1$ disproved such claim. This means that the existence of an upper bound specific to the real line is still an open question. This topic was part of the preliminary work for this thesis, and we believe such re-parametrization is possible, even though we couldn't prove it. Another potential research direction is the study of Permutation in other metric spaces of interest. In particular, the star metric could offer opportunities for alternative analysis.

References

[1] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

[2] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127(2):255–267, 1994.

[3] Bala Kalyanasundaram and Kirk Pruhs. On-line network optimization problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*, pages 268–280. Springer, 1996.

[4] Tsubasa Harada and Toshiya Itoh. A nearly optimal deterministic algorithm for online transportation problem. *CoRR*, abs/2406.03778, 2024.

[5] Abu Reyan Ahmed, Md. Saidur Rahman, and Stephen G. Kobourov. Online facility assignment. *Theor. Comput. Sci.*, 806:455–467, 2020.

[6] Tsubasa Harada, Toshiya Itoh, and Shuichi Miyazaki. Capacity-insensitive algorithms for online facility assignment problems on a line. *Discret. Math. Algorithms Appl.*, 16(5):2350057:1–2350057:39, 2024.

[7] Tsubasa Harada. A lower bound on the competitive ratio of the permutation algorithm for online facility assignment on a line. *CoRR*, abs/2402.12734, 2024.

[8] Enoch Peserico and Michele Scquizzato. Matching on the line admits no $o(\sqrt{\log n})$-competitive algorithm. *ACM Trans. Algorithms*, 19(3):28:1–28:4, 2023.

[9] Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 505–515, 2017.

[10] Sharath Raghvendra. Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[11] Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A $o(n)$-competitive deterministic algorithm for online matching on a line. *Algorithmica*, 81(7):2917–2933, 2019.

[12] Tsubasa Harada and Toshiya Itoh. Competitive analysis of online facility assignment for general layout of servers on a line. *CoRR*, abs/2308.05933, 2023.

[13] Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

[14] Bala Kalyanasundaram and Kirk Pruhs. The online transportation problem. *SIAM J. Discret. Math.*, 13(3):370–383, 2000.

[15] Barbara M. Anthony and Christine Chung. Online bottleneck matching. *J. Comb. Optim.*, 27(1):100–114, 2014.

[16] Varun Gupta, Ravishankar Krishnaswamy, and Sai Sandeep. Permutation strikes back: the power of recourse in online metric matching. In *Proceedings of the 23rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 40:1–40:20, 2020.

[17] Nicole Megow and Lukas Nölke. Online minimum cost matching with recourse on the line. In *Proceedings of the 23rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 37:1–37:16, 2020.

[18] Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Naor. A randomized $O(\log^2 k)$-competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.

[19] Bala Kalyanasundaram, Kirk Pruhs, and Clifford Stein. A randomized algorithm for online metric b-matching. *Oper. Res. Lett.*, 51(6):591–594, 2023.

[20] Kirk Pruhs. personal communication, 2024.

Acknowledgments

English version

The work for this thesis was a real adventure. It started many months ago, with a fascinating problem and now, through many unexpected discoveries, it came to an end. I must thank my supervisor, Prof. Michele Scquizzato, for his great idea of starting this work and his precious advices along the way. I had a lot of ups and downs but mostly I had fun, I thank him for giving me this opportunity and for everything I have learned. I would also like to thank Prof. Enoch Peserico who joined our work providing brilliant intuitions.

The journey that brought me to this point is only possible because of the support of my family: my parents Daniela and Alfredo and my brother Francesco. I thank them for always supporting my decisions and believing in me. They made everything possible with their hard work.

The person that I am today is the result of the the things I learned and the experiences I made. For the firsts, I want to thank the wonderful teachers I had in my years of study and name, from the long list, Prof. Mino Conte who left a big impact on me. For the experiences I have to thank all of my friends. The sociality and the sharing of fun and emotions helped me a lot. I am lucky enough to have them with me. Among them, Gianpiero and Giovanna deserve a special mention.

The last few years were among the bests of my life and I need to thank the University of Padova and the Math Department for this. But mostly I thank the many friends I met, and especially the few with whom I had lunch in mensa. Thank you for so many wonderful days.

The person I am most grateful to have met in Torre Archimede is my optimal matching Niloufar. Thank you for all the time and happiness you gave me.

Ringraziamenti

Italian version

Il lavoro per questa tesi è stato un' avventura. É iniziato molti mesi fa, con un problema affascinante e ora, dopo diverse scoperte inaspettate, è arrivato ad una conclusione. Devo ringraziare il mio relatore, il Prof. Michele Scquizzato, per la fantastica idea di iniziare questo lavoro e per i suoi preziosi consigli. Ho attraversato momenti alti e bassi, ma soprattuto mi sono divertito. Lo ringrazio per avermi dato questa opportunità e per tutto quello che ho imparato. Vorrei anche ringraziare il Prof. Enoch Peserico, che ha contribuito a questa tesi portando intuizioni brillanti.

Il viaggio che mi ha portato fino a questo punto è stato possibile solamente grazie al supporto della mia famiglia: i miei genitori Daniela e Alfredo e mio fratello Francesco. Li ringrazio per avere sepre supportato le mie decisioni ed aver creduto in me. Hanno reso tutto questo possibile grazie al loro duro lavoro.

La persona che sono oggi è il risultato delle cose che ho imparato e le esperienze che ho fatto. Per le prime, voglio ringraziare gli insegnanti meravigliosi che ho avuto nei miei anni di studi nominando, dalla lunga lista, il Prof. Mino Conte, che ha avuto un grande impatto su di me. Per le esperienze devo ringraziare tutti i miei amici e amiche. La socialità e la condivisione del divertimento e delle emozioni mi ha aiutato tanto. Sono fortunato ad averli con me. Tra di loro, Gianpiero e Giovanna meritano una menzione speciale.

Gli ultimi anni sono stati tra i migliori e devo ringraziare l'Università degli Studi di Padova e il Dipartimento di Matematica per questo. Ma soprattuto ringrazio i tanti amici che ho conosciuto, e specialmente i pochi con cui ho pranzato in mensa. Grazie per i tanti bei giorni trascorsi assieme.

La persona che sono più grato aver incontrato in Torre Archimede è il mio *optimal matching* Niloufar. Grazie per tutto il tempo e la felicità che mi hai donato.