The k-server problem

Elias Koutsoupias*

April 29, 2009

Abstract

The k-server problem is perhaps the most influential online problem: natural, crisp, with a surprising technical depth that manifests the richness of competitive analysis. The k-server conjecture, which was posed more than two decades ago when the problem was first studied within the competitive analysis framework, is still open and has been a major driving force for the development of the area online algorithms. This article surveys some major results for the k-server problem.

1 Introduction

The k-server problem in a metric space (or weighted graph) is the problem of efficiently moving around k servers to service requests that appear online at the points of the metric space. The k servers start at a fixed set of k points which we call the initial configuration. Then in each time step a request appears at some point of the metric space and some server has to move to that point to service the request. We assume that the requests appear sequentially and that a new request appears only after the current request has been serviced. The goal is to minimize the total distance traveled by the servers. More formally, the definition of the problem is as follows:

Let M be a metric space and let $d(a_1, a_2)$ denote the distance in M of points a_1 and a_2. The distance d is nonnegative, symmetric, and satisfies the triangle inequality. For simplicity, we allow distinct points to be at distance 0, or more formally, M is pseudometric rather than metric. Special spaces such as the uniform metric space (with all distances equal), the line (or Euclidean 1-dimensional metric space) and tree metric spaces are of particular interest for the k-server problem.

We are interested in sets of k points of M which we call *configurations*. Let $M^{(k)}$ denote the set of all configurations of M. We extend the notion of distance from M to $M^{(k)}$: If C_1, C_2 are configurations then $d(C_1, C_2)$ is the value of the minimum-weight perfect matching between the points of C_1 and C_2. In other words, it is the minimum distance traveled by k servers that change configuration from C_1 to C_2.

The k-server problem is defined by an *initial configuration* $C_0 \in M^{(k)}$ and a sequence of *requests* $r = (r_1, \ldots, r_m)$ of points of M. A solution is a sequence of configurations $C_1, \ldots, C_m \in M^{(k)}$ such that $r_t \in C_t$ for all $t = 1, \ldots, m$. The k servers start at configuration C_0 and service requests $r_1, \ldots, r_m$ by moving through configurations $C_1, \ldots, C_m$. The cost of a solution is the total distance traveled by the servers which is $\sum_{t=1}^{m} d(C_{t-1}, C_t)$. The objective is to find a solution with minimum cost.

An *online algorithm* computes each configuration C_t based only on the past and in particular on $r_1, \ldots, r_t$ and $C_0, \ldots, C_{t-1}$[1]. The central question in competitive analysis is how good is an

*University of Athens, elias@di.uoa.gr. Supported in part by IST-2008-215270 (FRONTS).

[1]Although for deterministic algorithms the configurations can be inferred from C_0 and the sequence of requests r, randomized algorithms are based on all configurations and in particular on C_{t-1}.

online algorithm compared to an optimal algorithm which is based also on the future requests $r_{t+1}, \ldots, r_m$. For convenience, we usually compare online algorithms to some other algorithm, not necessarily optimal, and we use the term *offline algorithm* to refer to it.

For an initial configuration C_0 and a sequence of requests $r = (r_1, \ldots, r_m)$, let $\mathrm{cost}_A(C_0, r)$ denote the cost of an online algorithm A and let $\mathrm{opt}(C_0, r)$ denote the cost of an optimal solution. We say that algorithm A has competitive ratio ρ, or that it is ρ-competitive, if for every C_0 and r

$$\mathrm{cost}_A(C_0, r) \leq \rho \cdot \mathrm{opt}(C_0, r) + \Phi(C_0), \tag{1}$$

for some $\Phi(C_0)$. We are interested in the competitive ratio ρ, which is essentially the approximation ratio of algorithm A. The "constant" term $\Phi(C_0)$, which is independent of the sequence of requests, is used to eliminate the dependency on the initial configuration C_0[2].

For a given metric space M, let ρ_M be the infimum of the competitive ratio among all online algorithms. We say that ρ_M is the competitive ratio of metric M. The competitive ratio of the problem itself is the worst-case (more precisely, the supremum) of the competitive ratio of all metric spaces.

Rather unexpectedly, it is very likely that the competitive ratio is independent of the metric space (provided that it has more than k distinct points). This is supported by all results on the problem in the last two decades and it is exactly what the k-server conjecture states:

Conjecture 1 (The k-server conjecture)**.** *For every metric space with more than k distinct points, the competitive ratio of the k-server problem is exactly k.*

The k-server conjecture is open, but it is known to be true for $k = 2$ and for some special metric spaces. For the general case, a slightly weaker result has been established [Kou94, KP95] for every metric space: The competitive ratio is at most $2k - 1$. We will discuss some of these results below.

When we analyze online problems, it is useful to view the situation as a game between us and an adversary. This is apparent from the definition of the minmax expression in the definition of the competitive ratio: the competitive ratio of a problem (if we ignore the additive constant term) is

$$\rho = \inf_A \sup_r \frac{\mathrm{cost}_A(C_0, r)}{\mathrm{opt}(C_0, r)},$$

where A ranges over all online algorithms and r over the possible sequences of requests. We can interpret this as follows: We, the online algorithm designers, come up with an online algorithm A, an adversary comes up with an input r and we pay to the adversary the ratio $\mathrm{cost}_A(r)/\mathrm{opt}(r)$. This worst-case analysis has a few drawbacks, the most prominent of which is that the adversary knows the online algorithm and can create a "nemesis" input specifically tailored for this online algorithm. To try to alleviate this advantage of the adversary, we consider *randomized online algorithms*. The adversary still knows the online algorithm but it does not know the outcome of the random choices. The most natural (and, from the point of view of the adversary, the most restrictive) assumption is that the adversary knows nothing about the random choices, so it might as well design the complete input once and for all at the beginning of the game. This is the *oblivious adversary* model. There are other models of adversaries against online algorithms in which the adversary gets information about some random choices, such as the online and the offline adaptive adversaries. The oblivious adversary has become the standard model for analyzing randomized online algorithms and we consider only this model here. We mention however that there are some interesting open problems regarding the online adaptive adversary [BEY98].

Perhaps the most challenging open problem about the k-server problem is captured by the following conjecture:

---

[2] We may allow the term $\Phi(C_0)$ to depend or not on the metric space and not just on the subspace spanned by C_0, but all known results indicate that this does not affect the competitive ratio.

Conjecture 2 (The randomized k-server conjecture)**.** *For every metric space, there is a randomized online algorithm for the k-server problem with competitive ratio $O(\log k)$.*

There has been important progress towards this conjecture [Bar96, BBBT97, BBM01], yet the situation is disappointing: for general metric spaces, we know of no better upper bound on the competitive ratio of randomized algorithms than the bound of deterministic algorithms (that is, $2k-1$). Even for the special case of $k = 2$, for which the deterministic case is completely settled 2 decades ago, the best known upper bound is 2.

2 Brief history of the problem

The problem was first defined by Manasse, McGeogh, and Sleator in [MMS88]. The definition of the problem was the result of important recent developments in online algorithms. In particular, a few years before, Tarjan and Sleator [ST85a] initiated the study of online algorithms using the framework of competitive analysis—although they called it "amortized efficiency" at the time—and they analyzed algorithms for the paging problem and the list update problem. Karlin, Manasse, Rudolph, and Sleator [KMRS86, KMRS88] used the framework to study snoopy caching (a caching technique for multiprocessors that share a common bus), introduced the term competitive analysis, and raised the question of applying competitive analysis to other online problems. A great step forward to study these questions in a more general setting was made by Borodin, Linial, and Saks [BLS87, BLS92] who introduced the online problem of metrical task systems, a generalization of the k-server problem. They succeeded in determining exactly the competitive ratio: it is $2n - 1$ for a metrical task system of n states. This was the situation when Manasse, McGeogh, and Sleator introduced the k-server problem. It is a special case of the metrical task systems problem, in the sense that it can be expressed as a metrical task system in which only certain requests are allowed. It is also a great generalization of the paging problem; the paging problem is essentially the special case of the k-server problem in uniform metric spaces. By lucky coincidence the level of generality of the problem seems to be right below the threshold of the problems whose competitive ratio is independent of the size of the metric or the length of the sequence request. Manasse, McGeogh, and Sleator showed a few important results: They showed than no online algorithm can have competitive ratio less than k; this is independent of the metric space as long as the metric space has at least $k + 1$ points. Furthermore, they showed that the competitive ratio is exactly 2 for the special case of $k = 2$, and it is exactly k for all metric spaces with $k + 1$ points. With this weak evidence, they boldly posed the k-server conjecture.

More evidence for the k-server conjecture appeared soon. On the one hand, computer experiments on some small metric spaces verified the conjecture for $k = 3$. On the other hand, the conjecture was shown to hold for the line (the 1-dimensional Euclidean space) [CKPV91] and for tree metric spaces [CL91b].

Online algorithms became a very active research area around 1990 [BEY98, FW98]. Besides the problems that relate to the k-server problem there was also research in other areas such as online search and exploration problems [BYCR88, PY89, PY91b], online data structures [ST85b], scheduling [GW93, BFKV92], and many others.

An important early result about the general k-server problem was shown by Rabani, Ravid, and Fiat [FRR90]. They gave an online algorithm with competitive ratio bounded by a function of k; however the bound was exponential on k. This was improved by combining a result of Grove [Gro91] with the derandomization technique of [BDBK$^+$94] to $O(k^2 4^k)$. A dramatic improvement was shown in [KP94], which established that a natural online algorithm, the Work Function Algorithm, has competitive ratio at most $2k - 1$. This remains the best known bound. Since this 1994 result, there has been limited progress on the k-server conjecture. The conjecture was established for the special case of metrics with $k + 2$ points in [KP96] (in fact, this result appeared together with the $2k-1$ bound in [Kou94]). For the general case, a second proof of the $2k-1$ was given in [Kou99]. The same paper extended the $2k-1$ bound to the case in which the online algorithm competes against offline algorithms that have less than k servers (weak adversaries).

There are 2 special cases of the k-server problem that have been studied and are of particular interest: the 3-server problem[CL94, BCL02] and the k-server problem on a cycle [FRRS91]. Any progress on these problems has the potential to open new paths to attack the general k-server conjecture. For both of these special cases, we know nothing better than the $2k-1$ bound.

The most important recent developments for the k-server problem are about randomized algorithms. The history of randomized algorithms for the k-server problem goes back to the end of the 1980s. Raghavan and Snir [RS89] were the first to study the Harmonic Algorithm, a natural memoryless algorithm that moves its servers with probability inversely proportional to the distance. They showed that for metric spaces of $k+1$ points the competitive ratio of the Harmonic Algorithm is at most $k(k+1)/2$ and that this is the best possible. Unlike the known results for the deterministic case, the competitive ratio of randomized algorithms is not the same for all metric spaces. For example, the Harmonic Algorithm has competitive ratio k for the uniform metric space of $k+1$ points but it has competitive ratio almost $k(k+1)/2$ for the uniform metric space in which we contracted a single edge to some arbitrarily small ϵ. Berman, Karloff, and Tardos [BKT90] showed that the Harmonic Algorithm has bounded competitive ratio for $k=3$. This was followed by the important result of Grove [Gro91] who showed that the Harmonic Algorithm has competitive ratio $O(k2^k)$. At about the same time, Ben-David, Borodin, Karp, Tardos, and Wigderson [BDBK$^+$90] studied the power of randomization in online algorithms. They established that there is a way to derandomize a memoryless online algorithm with competitive ratio ρ to get a deterministic algorithm with competitive ratio at most ρ^2. Combined with the result of Grove, this implied that the deterministic competitive ratio for the k-server problem is $O(k^2 4^k)$. The result of Grove was later improved slightly and simplified by Bartal and Grove [BG00]. For the special case of $k=2$ servers, Chrobak and Larmore [CL92a] showed that the exact ratio of the Harmonic Algorithm is at most 3 on every metric space.

The Harmonic Algorithm is a very natural memoryless algorithm but it is not the best one for metrics of $k+1$ points. For this important special case of metric spaces, it was shown by Coppersmith, Doyle, Raghavan, and Snir [CDRS93] that there is a memoryless randomized algorithm for $k+1$ points which has competitive ratio k, by exploiting the connection of the theory of electrical networks with random walks. This is the best possible competitive ratio for memoryless algorithms.

The next important phase of developments about the k-server problem involved randomized algorithms with memory. In the late 1980s, Karp in an unpublished work, introduced the idea of probabilistic embedding of metrics. In particular, he showed that a cycle can be probabilistically approximated by the line within a factor of 2. This allows any online algorithm on the line to be transformed into a randomized algorithm for the cycle with at most double competitive ratio. This idea was extended by Alon, Karp, Peleg, and West [AKPW92] and it was generalized by Bartal [Bar96] to every metric space. Bartal introduced the notion of hierarchically well-separated trees (HST) which are special types of hierarchical trees that facilitate a divide-and-conquer approach. He showed that every metric can be probabilistically approximated by an HST with polylogarithmic distortion. This was improved to $O(\log n \log\log n)$ distortion in [Bar98] and to the optimal $O(\log n)$ distortion in [FRT04].

Using such probabilistic embeddings, it was shown by Bartal, Blum, Burch, and Tomkins [BBBT97] that for metric spaces with $k+c$ points, where c is a constant, the randomized competitive ratio is polylogarithmic in k: $O(\log^5 k \log\log k)$. This was improved substantially by Fiat and Mendel to $O((\log k \log\log k)^2)$ [FM03]. A more recent result by Cote, Meyerson, and

Poplawski [CMP08] gave an algorithm for metric spaces that can be hierarchically approximated by binary trees; the algorithm has competitive ratio $O(\log \Delta)$, where Δ is the diameter of the metric space.

For uniform metric spaces, the k-server problem is equivalent to the paging problem. For this interesting special case, it was shown by Fiat, Karp, Luby, McGeoch, Sleator, and Young [FKL$^+$91] that the randomized competitive ratio is between $H_k = 1 + 1/2 + 1/3 + \ldots + 1/k$ and $2H_k$ by analyzing the Mark Algorithm. This algorithm was shown later in [ACN00] to have slightly better competitive ratio: $2H_k - 1$. McGeogh and Sleator settled the competitive ratio for uniform metric spaces, by giving an algorithm which is H_k-competitive. A different k-competitive algorithm was given later by Achlioptas, Chrobak, and Noga [ACN00], exploiting a characterization of work functions in [KP00].

For the weighted paging problem (equivalent to the k-server problem on stars where the requests are at the end of leaves), Irani [Ira97] established the randomized k-server conjecture for the weighted paging with 2 weights. This was generalized to every number of weights by Bansal, Buchbinder, and Naor [BBN07].

For general metric spaces, it was pointed out in [KMMO94] that there are metric spaces that have higher competitive ratio than H_k, even for $k = 2$. On the other hand, it is still open whether there are metric spaces of $k + 1$ distinct points with competitive ratio strictly less than H_k. The best known lower bound for every metric space is $\Omega(\log k / \log \log k)$, due to Bartal, Bollobás, and Mendel [BBM01] and Bartal, Linial, Mendel and Naor [BLMN03]. They improved a bound $\Omega(\sqrt{\log k / \log \log k})$ due to Blum, Karloff, Rabani, and Saks [BKRS92].

Research on the k-server problem progressed in parallel with developments on other similar problems. Most notably, the problem of *metrical task systems*, a generalization of the k-server problem, for which [BLS92] gave tight upper and lower bounds for deterministic algorithms. It was shown later that the Work Function Algorithm has also optimal competitive ratio for this problem. For randomized algorithms with memory, the two problems are intertwined. Another related problem that is more general than the k-server problem and a special case of metrical task systems is the metrical service systems [CL92b, CL92c] which is equivalent to the layered graph traversal problem [FFK$^+$91, Bur94]. Also the CNN problem [KT00, SSdP03] is similar to the k-server problem, especially to the weighted version in which the servers have different speeds (or costs) [CS04].

2.1 Outline

We first consider deterministic algorithms and then randomized algorithms. For deterministic algorithms, we start with an easy lower bound of k and then discuss tree metric spaces, an important special case of the problem. We discuss the Double Coverage Algorithm which is a natural algorithm and whose analysis is typical for the k-server problem. We then proceed to discuss the Work Function Algorithm and the structure of the proof that it is $(2k - 1)$-competitive. The discussion of the special case of metrics with $k + 1$ points provides a useful derivation from deterministic to randomized algorithms. We then discuss briefly randomized algorithms, and in particular, the Harmonic Algorithm.

3 Deterministic algorithms

In this section we mainly discuss a lower bound for deterministic online algorithms and two important algorithms: the Double Coverage Algorithm on tree metrics and the Work Function Algorithm on all metrics.

3.1 Lower bound

The following result was established by [MMS88] when they introduced the k-server problem.

Theorem 1. *In every metric space with at least $k+1$ points, no online algorithm for the k-server problem can have competitive ratio less than k.*

To prove it, let A be an online algorithm. We want to show that there are request sequences of arbitrarily high cost for A for which the online algorithm A has cost k times more than the optimal algorithm, within some additive constant term.

The proof is based on a simple trick: Instead of comparing algorithm A against one offline algorithm, we compare its cost against k distinct offline algorithms so that *the cost of the online algorithm is equal to the total cost of all k offline algorithms.* First we fix a subspace of $k+1$ points that include the original configuration. All requests are going to be in this subspace. The nice thing about spaces of $k+1$ points is that we know precisely what an optimal adversary does: at any point in time, there is only one point of the metric space that the online algorithm does not cover and the adversary should request it next. By repeating this enough times, the online cost can become arbitrarily high.

There are exactly $k+1$ configurations of the subspace of $k+1$ points. We consider k offline algorithms $B_1, \ldots, B_k$ and we maintain the following invariant: The configurations of algorithm A and of the k algorithms $B_1, \ldots, B_k$ are all different (and thus they are all the $k+1$ possible configurations of the subspace). To achieve the invariant initially, we have to move the servers of the offline algorithms from the original configuration to all other configurations. This is a fixed cost that depends only on the original configuration. Suppose now that at time t the online algorithm A moves a server from some point a to the request r_t. All k offline algorithms cover the request r_t and they do not have to move. However, one of them makes a non-lazy move in order to maintain the invariant. In particular, when A moves its server from a to r_t, the new configuration of algorithm A becomes identical to the configuration of some offline algorithm B_i. Algorithm B_i then moves one of its servers in the opposite direction from r_t to a so that its new configuration matches the previous configuration of A. The other offline algorithms do not move. Thus the invariant is maintained. The cost of A is $d(a, r_t)$ which is equal to the total cost of all offline algorithms, $d(r_t, a)$.

In summary, apart from the initial moves of the offline servers, the total cost of the online algorithm is exactly equal to the cost of k offline algorithms, which shows that the competitive ratio is at least k.

3.2 Deterministic algorithms for tree metrics

An important special case of the k-server problem is the case of the 1-dimensional Euclidean space. This is an abstraction of many applications, most typical of which is the k-head disk problem: Each of the k heads (which correspond to servers) of a hard disk can move freely along a straight line to be positioned above a track for reading or writing.

This special case of the problem is also important because it involves interesting algorithms and analysis. A very elegant online algorithm for the line is the *Double Coverage Algorithm.* It is like the Greedy Algorithm which always moves the closest server to the request. Instead, the Double Coverage Algorithm usually moves the two closest servers towards the request. In particular, when the request is between two servers, the algorithm moves both servers towards the request; the servers travel the same distance, equal to the distance of the request to the closest of the two servers. When the request is outside the interval of the k servers, only the closest server moves to it.

The definition of the Double Coverage Algorithm brings to the fore the important issues of *laziness* and *memorylessness.* Because we are dealing with metric spaces, there are solutions of minimum cost in which consecutive configurations differ only in one point and more precisely $C_t \subset C_{t-1} \cup \{r_t\}$. We call such solutions *lazy,* since they correspond to moving servers only to service directly requests. Most natural algorithms are lazy, but there are important examples of non-lazy algorithms, and in particular the Double Coverage Algorithm. A non-lazy algorithm may move its servers to place them in better positions for future requests. However, non-lazy algorithms have no real advantage in metric spaces: in fact, every non-lazy algorithm can be simulated by a lazy algorithm by postponing the non-lazy moves until they become necessary (and thus lazy). In the process, some non-lazy moves may be combined into one lazy move (when for example a server moves non-lazily from a to b and later from b to c, the lazy move will go directly from a to c); because of the triangle inequality, this can only lower the cost.

An important aspect of some natural algorithms is that they are memoryless. An algorithm is called memoryless when it decides how to service a request based only on the current configuration. One has to be careful with this concept when we allow the servers to move freely, because in some metric spaces (such as the Euclidean), with negligible cost, the algorithm can slightly move the servers to encode the whole history in the current configuration. To avoid this pitfall, we insist on memoryless algorithms being lazy.

Double Coverage Algorithm is not a lazy algorithm, but it can be turned into one by postponing and combining some moves. It is a memoryless algorithm, in the sense that the decisions are based only on the current configuration. However, as we mentioned, we have to be careful with the issue of memorylesness when one encodes, with negligible cost, the whole past by perturbing appropriately the current configuration.

The analysis of the Double Coverage Algorithm is a typical application of the *potential function method*. To use this method, we define a potential $\Phi(C_t, C'_t)$ which captures the disadvantage of the online configuration C_t with respect to the offline configuration C'_t. If cost_t and opt_t denote the online and offline costs to service the request r_t, then we use the potential to show that

$$cost_t - \rho \cdot \mathrm{opt}_t \leq \Phi(C_{t-1}, C'_{t-1}) - \Phi(C_t, C'_t)$$

If we can establish this, then by adding all steps t we get that

$$\sum_{t=1}^{m} \mathrm{cost}_t - \rho \cdot \sum_{t=1}^{m} \mathrm{opt}_t \leq \Phi(C_0, C'_0)$$

which shows that the competitive ratio is at most ρ. In fact, the potential function method is nothing more than the technique of strengthening the induction hypothesis.

We have only fragments of a theory to guide us to find an appropriate potential function Φ, but fortunately, for the Double Coverage Algorithm a very natural potential function works. It consists of 2 terms: The first term is $k \cdot d(C_t, C'_t)$ and the reasoning behind it is that when the adversary moves an offline server some distance x, the distance $d(C_t, C'_t)$ increases by at most x and the online algorithm will eventually pay back k times more this cost. The second term is less obvious but it plays an important role even in the general case of the k-server problem; it is not specific to the Double Coverage Algorithm but to every online algorithm. This term is equal to the sum of all pairwise distances between the servers of the online configuration C_t. The reasoning behind this term is that when the offline configuration contracts, the online algorithm pays more than k times the distance; it is easier to explain the situation when $k = 2$. Consider an initial configuration of two servers consisting of points a and b. The adversary puts a request at the point c in the middle of the interval $[a, b]$. By symmetry, we can assume that the online algorithm services the request at c using the server at a. The offline algorithm services c using the other server at b. Now a sequence of requests at a and c forces any reasonable online algorithm to move to the configuration $\{a, c\}$. The total cost of the online algorithm is at least $3d(b, c)$ while the offline cost is only $d(b, c)$. This suggests a competitive ratio of 3 for $k = 2$ servers. However, the online algorithm “gained” something by bringing its servers closer. This gain is captured by the second term of the potential function.

We define $\Phi(C_t, C'_t)$ precisely:

$$\Phi(C_t, C'_t) = k \cdot d(C_t, C'_t) + \sum_{a_1, a_2 \in C_t} d(a_1, a_2).$$

The proof that the Double Coverage Algorithm is k-competitive is based on the observation that when an offline server is already at the request, the double move of the online algorithm guarantees that the first term of the potential function does not increase, because one of the two moving servers moves towards its matching point of the offline configuration; although the other online server may move away from its matching offline server an equal distance, their total contribution does not increase the matching. The other term of the potential (the sum of distances between the online servers) decreases by the total distance traveled by the two servers. More precisely, let s_i and s_j be the positions of the 2 online servers closest to the request r_t with $d(s_i, r_t) \leq d(s_j, r_t)$. The two servers move towards r_t; this decreases the potential function by $2d(s_i, r_t)$.

Let C_t, C'_t be the online and offline configurations after serving request r_t. The above observation implies

$$\Phi(C_t, C'_t) \leq \Phi(C_{t-1}, C'_t) - 2d(s_i, r_t) = \Phi(C_{t-1}, C'_t) - d(C_{t-1}, C_t)$$

The same inequality

$$\Phi(C_t, C'_t) \leq \Phi(C_{t-1}, C'_t) - d(C_{t-1}, C_t) \tag{2}$$

holds also when the request r_t is outside the interval of the k servers and only one server moves from s_i to the request r_t. Here, the first term of the potential decreases by $k \cdot d(s_i, r_t)$, because the online server moves closer to its matching point and the second term increases by $(k-1) \cdot d(s_i, r_t)$. The difference is $d(s_i, r_t)$, equal to the online cost $d(C_{t-1}, C_t)$.

The above hold for C'_t when an offline server is already at the request r_t. We have also to take into account the moves of the offline algorithm. By the definition of Φ and the properties of the metric space, we get

$$\Phi(C_{t-1}, C'_t) \leq \Phi(C_{t-1}, C'_{t-1}) + k \cdot d(C'_{t-1}, C'_t) \tag{3}$$

Combining (2) and (3), we get

$$d(C_{t-1}, C_t) - k \cdot d(C'_{t-1}, C'_t) \leq \Phi(C_{t-1}, C'_{t-1}) - \Phi(C_t, C'_t)$$

which implies a competitive ratio k for the Double Coverage Algorithm.

One nice property of the Double Coverage Algorithm is that it can be extended directly to tree metric spaces. To describe the Double Coverage Algorithm for tree metric spaces, we say that a server s_i is "unobstructed" if there is no other server in the unique path from s_i to the current request r_t.

Double Coverage Algorithm for trees To service request r_t, move all unobstructed servers towards r_t with equal speed. When during this process, a server becomes obstructed, it stops while the rest keep moving.

The algorithm is k-competitive for every tree metric space. The proof is almost identical to the proof above for the 1-dimensional Euclidean space.

Can we extend these ideas to other cases? The answer is positive for $k = 2$ servers and any metric space. The underlying reason is that every metric space of 3 points is a tree metric space (in the sense that it can be isometrically embedded into a tree). In particular, the configuration $\{a_1, a_2\}$ of the 2 servers and the next request r_t induce a metric subspace of 3 points which can be embedded into a tree as in Figure 1. Imagine that we extend the original metric space by adding the 3-star of the figure, so that we can apply the Double Coverage Algorithm in this new metric space. There are two hurdles to clear. First, the new space is not a tree. This is easy to solve: The potential argument of the Double Coverage Algorithm applies to it as well. Second, the Double Coverage Algorithm may move a server to some point of the added 3-star, which is not a point of the original metric space. The solution to this is also easy: By transforming the

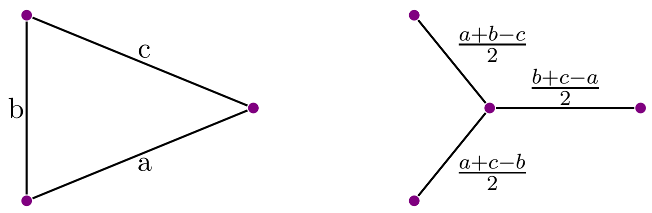

Figure 1: The 3-point metric on the left is equivalent to the star on the right.

Double Coverage Algorithm to a lazy algorithm, we guarantee that all servers move to requests, which are points of the original metric space. The scaffold of the added 3-stars (one for each request) is used only for bookkeeping by the lazy algorithm.

What about the case $k \geq 3$? Unfortunately, the approach cannot be extended to this case, as there are metric spaces of 4 points (3 servers plus the request) that are not trees; for example, the cycle of equally-spaced points (for a more general view of these issues see [LO06]). It is however true that every 4-point metric space is an L_1-metric. One can try to extend these ideas to other types of metrics, in particular to L_1-metrics, and this approach had some partial success (see for example [CL94, LO06]). An interesting question is whether the general k-server problem is easier on L_1-metrics. Relevant to this question is a result that the k-server conjecture holds for the $k = 3$ in the Manhattan plane[BCL02].

The Work Function Algorithm, which we discuss in the next section, is k-competitive on the line [BK04] and the proof is based on a potential argument, as well. However, it remains a challenging open problem whether the Work Function Algorithm is k-competitive for trees. Resolving this may prove helpful for the general case.

Open Problem. Show that Work Function Algorithm is k-competitive for trees.

3.3 The Work Function Algorithm

What is the computational complexity of the offline k-server problem? We describe a dynamic programming approach which is the basis of the Work Function Algorithm, a fundamental online algorithm.

To compute the optimal solution, we can safely restrict our attention to the submetric spanned by the initial configuration and the set of requests. For every configuration X, we define $w(C_0; r_1, \ldots, t_t; X)$ to be the cost of the optimal solution which starts at the initial configuration C_0, passes through points $r_1, \ldots, r_t$ (in that order) and ends up at configuration X. More formally, it is

$$w(C_0; r_1, \ldots, r_t; X) = \min\{\sum_{i=1}^{t+1} d(C_{i-1}, C_i) \,:\, C_i \in M^{(k)} \text{ with } r_i \in C_i, \text{ for } i = 1, \ldots, t \text{ and } C_{t+1} = X\}. \tag{4}$$

For fixed C_0 and $r_1, \ldots, r_t$, w is a real function of $M^{(k)}$. We call such a function *work function* and denote it as $w_{C_0; r_1, \ldots, r_t}$ or simply w_t, when the parameters C_0 and r are fixed. It is easy to compute the value of the work function $w_t(X) = w(C_0; r_1, \ldots, r_t; X)$ in a dynamic programming manner. We compute for all times $i = 1, \ldots, t$

$$w_i(X) = \min\{w_{i-1}(Z) + d(Z, X) \,:\, Z \in M^{(k)} \text{ with } r_i \in Z\}$$

using the bootstrapping values $w_0(X) = d(C_0, X)$.

We say that a configuration X is not in the *support* of work function w_t, when $w_t(X) = w_t(Z) + d(Z, X)$, where Z is a different configuration with $d(Z, X) > 0$. The support of a work function w_t is the minimal set of configurations that completely determines the value of

w_t. The work function of configurations X outside the support can be computed by $w_t(X) = \min_Z\{w_t(Z) + d(Z, X)\}$, where Z ranges over all configurations in the support. Intuitively, the configurations in the support are preferable to algorithms that prefer to pay later than sooner.

The most natural online algorithm is of course the Greedy Algorithm which services every request by the closest server. But is has unbounded competitive ratio because in some cases even when there is only one configuration in the support, it fails to move its servers to this configuration (for example, when an online server is far away); it therefore incurs unbounded cost while the offline algorithm occupies the configuration in the support and does not move at all. The other extreme is the Retrospective Algorithm which chases the best configuration: in every step, it moves to the configuration which minimizes the current work function w_t. It is not hard to see that Retrospective Algorithm also has unbounded competitive ratio.

The Work Function Algorithm balances the two extremes of the Retrospective Algorithm and the Greedy Algorithm.

Definition 1. To service the request r_t, the Work Function Algorithm moves from configuration C_{t-1} to a configuration C_t which contains r_t and minimizes

$$w_t(C_t) + d(C_{t-1}, C_t).$$

To understand and analyze the Work Function Algorithm we need to understand the properties of work functions. A handy notation is to represent by $X - a + b$ the configuration that results when in configuration X we replace point a by point b.

Some obvious properties of work functions are:

1. If $r_t \in X$, then $w_t(X) = w_{t-1}(X)$; that is, when an optimal algorithm is already at request r_t, there is no additional cost to service r_t. If $r_t \notin X$, then $w_t(X) = w_t(X-x+r_t)+d(r_t, x)$ for some $x \in X$; which means that to end up at configuration X, a server moves from x to r_t. Combining the two cases together, we get that for every configuration X:

$$w_t(X) = \min_{x\in X}\{w_{t-1}(X - x + r_t) + d(r_t, x)\},$$

which gives a simple way to compute by dynamic programming the values of work functions.

2. For every configurations X and Y: $w_t(X) \le w_t(Y) + d(X, Y)$. This shows that as the values of the work function increase, they remain close (at least for bounded metric spaces). This provides the following intuition: think of the values of the work function as almost equal to the cost of the optimal algorithm.

3. For every configuration X: $w_t(X) \ge w_{t-1}(X)$. Because $w_t(X) = w_{t-1}(X-x+r_t)+d(r_t, x)$, for some x and by the previous property $w_{t-1}(X) \le w_{t-1}(X - x + r_t) + d(r_t, x)$.

Are the previous properties enough to characterize work functions? Or to put it in another way, if a function w_t satisfies these properties, does it mean that there is an initial configuration C_0 and a sequence of requests r, for which $w_t(X) = w_t(C_0; r_1, \ldots, r_t; X)$ for every X? Not necessarily. The previous properties miss an important aspect of work functions, the *quasiconvexity property*.

Definition 2 (Quasiconvexity)**.** For every configurations X and Y, there is a bijection $h : X \to Y$ such that for every partition of X into X_1 and X_2:

$$w_t(X) + w_t(Y) \ge w_t(X_1 \cup h(X_2)) + w_t(h(X_1) + X_2).$$

There exists a simple graphical proof of the quasiconvexity property of work functions. Consider the k directed paths that start at the initial configuration C_0, pass through the requests $r_1, \ldots, r_t$ (in the given order) and finish at the points of configuration X. $w_t(X)$ is, by definition, the minimum weight of such paths. We call the edges of these paths X-edges and we color them blue. An obvious but crucial property of the blue edges, is that every point in r has indegree 1 and outdegree 1, every point in C_0 has outdegree 1, and every point in X has indegree 1. Similarly, there are k directed paths that finish at the points of configuration Y. We call these Y-edges and color them red (see Figure 2).

We now describe a simple parity argument which gives a bijection h between the points of X and Y.

> Let x be a point of X. We start at x and follow backward the unique blue X-edge that points to x. Then, we follow forward the unique red Y-edge, and we keep repeating this process following alternatively X-edges backwards and Y-edges forward. We also change the colors of the edges that we traverse.

The process follows a unique path that unavoidably will end at a point $y \in Y$; we set $h(x) = y$. It is a simple observation that if the process starts at some other point $x' \in X$, it will end at a different point $y' \in Y$. This shows that h is a bijection.

To show the quasiconvexity inequality, we apply the above process starting at each point in X_2. At the beginning, the blue edges form k paths that end at configuration X. After the changes, the blue edges form k paths that end at configuration $X' = X_1 \cup h(X_2)$. These k paths have length at least $w_t(X')$ because they form a feasible but not necessarily optimal solution for $w_t(X')$. Similarly the red edges at the beginning form k paths that end at configuration Y and at the end form k paths that end at configuration $Y' = h(X_1) \cup X_2$. The new red paths have length at least $w_t(Y')$. The process changed only the color of the edges, not their length. At the beginning the total length of blue and red edges is $w_t(X) + w_t(Y)$ and at the end the total length of the blue and red edges is at least $w_t(X') + w_t(Y')$, which shows the quasiconvexity inequality. An example is given in Figure 2.

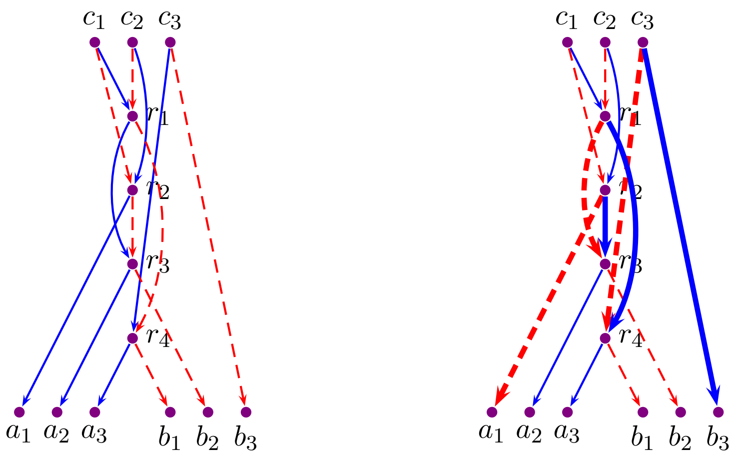

Figure 2: Illustrating the quasiconvexity property: Starting at a_1, we move backward to r_2, then forward to r_3, and so on, until we end up at b_3. The total length of all lines on the left is $w_4(A) + w_4(B)$ and on the right at least $w_4(A - a_1 + b_3) + w_4(B - b_3 + a_1)$.

Interestingly, we do not need the fact that M is a metric space in the above argument.

We now discuss a more complicated property of work functions, the *duality property* which involves work functions at consecutive steps $t-1$ and t. While quasiconvexity is an important property of work functions it is not used directly to prove the competitiveness of the Work Function Algorithm; instead it is used only to establish the duality property. It is the duality property that plays central role in the proof of competitiveness.

To describe it, let us define $d(r_t^k, X)$ to be the distance of all points of X from r_t (r_t^k is like a configuration with k copies of r_t).

Lemma 1 (Duality property). *Let A be a configuration which minimizes the expression $w_{t-1}(X) - d(r_t^k, X)$ over all configurations X. Then A minimizes also the expression $w_t(X) - d(r_t^k, X)$ and maximizes the expression $w_t(X) - w_{t-1}(X)$. That is*

$$\begin{aligned} & A \subseteq \operatorname{argmin}_X\{w_{t-1}(X) - d(r_t^k, X)\} \\ \Rightarrow \quad & A \subseteq \operatorname{argmin}_X\{w_t(X) - d(r_t^k, X)\} \\ \textit{and} \quad & A \subseteq \operatorname{argmax}_X\{w_t(X) - w_{t-1}(X)\} \end{aligned}$$

A configuration in $\operatorname{argmin}_X\{w_{t-1}(X) - d(r_t^k, X)\}$ is called a *minimizer* of r_t with respect to w_{t-1}. A direct consequence of the duality lemma is that

$$\max_X\{w_t(X) - w_{t-1}(X)\} = \min_X\{w_t(X) - d(r_t^k, X)\} - \min_X\{w_{t-1}(X) - d(r_t^k, X)\} \tag{5}$$

Although there is no clear intuition why the duality property holds, its proof follows from the quasiconvexity property or by similar, albeit less straightforward, graphical parity argument (for an algebraic derivation of the quasiconvexity and the duality property see [KP95]).

The duality property is useful because of the following important lemma:

Lemma 2 (Extended Cost Lemma). *If for every instance of the k-server problem*

$$\sum_{t=1}^{m} \max\{w_t(X) - w_{t-1}(X)\} \leq \rho \min_X w_m(X) + c, \tag{6}$$

for some constant term c which does not depend on the sequence of the requests, then the Work Function Algorithm is $(\rho - 1)$-competitive.

This lemma has a very important consequence: To prove the k-server conjecture, one needs only to prove (6) for $\rho = k + 1$, which is a property of work functions that does not involve the configurations of the online algorithm at all. Notice also how this lemma reveals the importance of the duality property which characterizes exactly the configurations that maximize the terms $\max\{w_t(X) - w_{t-1}(X)\}$ that we need to bound in the Extended Cost Lemma.

The proof of the Extended Cost Lemma (due to Chrobak and Larmore[CL92d]) is a direct consequence of the definition of the Work Function Algorithm: By definition, the configuration C_t of the Work Function Algorithm at time t is a configuration which contains r_t and minimizes $w_t(X) + d(C_{t-1}, X)$, or equivalently $w_{t-1} + d(C_{t-1}, X)$ as X contains r_t. But by the definition of work functions, the value of the work function $w_t(C_{t-1})$ is computed with exactly the same expression: $w_t(C_{t-1}) = \min_{X : r_t \in X}\{w_{t-1}(X) + d(X, C_{t-1})\}$. Therefore we get an important property of the Work Function Algorithm:

$$w_t(C_{t-1}) = w_{t-1}(C_t) + d(C_{t-1}, C_t). \tag{7}$$

This is a property which characterizes the Work Function Algorithm and provides some intuition on how the Work Function Algorithm works: The online algorithm tries to mimic the lower bound situation in Section 3.1 and moves in the opposite direction of some offline algorithm. In particular, to service the request r_t, the Work Function Algorithm moves from C_{t-1} to C_t, while an optimal algorithm services the request r_t and moves in the opposite direction, from C_t to C_{t-1}.

The total cost of the Work Function Algorithm at step t is $d(C_{t-1}, C_t) = w_t(C_{t-1}) - w_{t-1}(C_t) = w_t(C_{t-1}) - w_{t-1}(C_{t-1}) - (w_t(C_t) - w_{t-1}(C_{t-1}))$. We add and subtract the term $w_{t-1}(C_{t-1})$ because the second term telescopes (i.e., if we take the sum over t, the intermediate terms cancel out). For the first term $w_t(C_{t-1}) - w_{t-1}(C_{t-1})$, Chrobak and Larmore took a bold step and bounded it by $\max_X\{w_t(X) - w_t(X)\}$. With this, we get that the total cost of the Work Function Algorithm is:

$$\begin{aligned}\sum_{t=1}^{m} d(C_{t-1}, C_t) &\leq \sum_{t=1}^{m} (w_t(C_{t-1}) - w_{t-1}(C_{t-1}) - (w_t(C_t) - w_{t-1}(C_{t-1}))) \\ &\leq \sum_{t=1}^{m} \max_X\{w_t(X) - w_t(X)\} - (w_m(C_m) - w_0(C_0)).\end{aligned}$$

Without loss of generality, we can assume that at the end there is only one configuration in the support and that the Work Function Algorithm is at this configuration. This is because the adversary can, at the end, place many requests to the configuration of the offline algorithm; this will not increase the offline cost, while the Work Function Algorithm will incur some cost until it converges to this configuration. This means that $w_m(C_m) = \min_X w_m(X)$, and we see that the Extended Cost Lemma is true.

The duality property together with the Extended Cost Lemma gives us a firm ground to prove the bounded competitive ratio of the Work Function Algorithm. We need to find a way to bound $\max_X\{w_t(X) - w_{t-1}(X)\}$, or by using the duality property to bound $\min_X\{w_t(X) - d(r_t^k, X)\} - \min_X\{w_{t-1}(X) - d(r_t^k, X)\}$. Notice that the second min involves r_t and not r_{t-1} (and therefore this expression does not telescope when we sum over t). To show that the Work Function Algorithm is $(\rho - 1)$-competitive we need only to show that if we sum this expression over t, it is bounded above by $\rho \min_X w_m(X) + c$, where c is independent of the sequence of requests:

$$\sum_{t=1}^{m} \min_X\{w_t(X) - d(r_t^k, X)\} - \sum_{t=1}^{m} \min_X\{w_{t-1}(X) - d(r_t^k, X)\} \leq \rho \min_X w_m(X) + c. \tag{8}$$

The general idea for bounding the competitive ratio of the Work Function Algorithm is based on a potential argument. Such a typical argument runs as follows: Let

$$\psi_t(A_0, b_0, b_1 \ldots, b_p) = w_t(A_0) - d(b_0^k, A_0) + \chi_t(b_0, b_1, \ldots, b_n)$$

be an expression where $\chi_t(b_0, b_1, \ldots, b_n)$ is the sum of the value of the work function w_t on some configurations (each one consisting of points in $\{b_0, \ldots, b_p\}$) plus an arithmetic expression involving some distances of the same points. We illustrate the ideas by an example which we will employ to prove that Work Function Algorithm is 2-competitive for $k = 2$. We will use the expression $\psi_t(A_0, b_0, b_1, b_2) = w_t(A_0) - d(b_0^2, A_0) + w_t(\{b_0, b_1\}) + w_t(\{b_0, b_2\}) - d(b_1, b_2)$.

Notice the similarity of the terms that we want to bound in (8) with the first part of ψ_t, but pay attention to the fact that instead of r_t we have b_0. The role of χ_t is less obvious, but think of it as some premium that we have to pay to get a desired property of ψ_t. What is this desired property? *We want b_0 to be equal to r_t when we minimize ψ_t.* More precisely, when we take the minimum of ψ_t over all possible points of the metric space, this minimum should be achieved when b_0 is the last request r_t. This is the crucial technical step in the proof; it requires that we must establish

$$\min_{A_0, b_0, \ldots, b_p} \psi_t(A_0, b_0, \ldots, b_p) = \min_{A_0}\{w_t(A_0) - d(r_t^k, A_0)\} + \min_{b_1, \ldots, b_p} \chi_t(r_t, b_1, \ldots, b_p\}.$$

Once we have established the above, we can easily show that the competitive ratio of the Work Function Algorithm is equal to the number of values of the work function that appear in the expression χ_t. The reason is this: Take $A_0^*, b_0^* = r_t, b_1^*, \ldots, b_p^*$ that minimize ψ_t. The minimum of ψ_{t-1} is at most $\min_{A_0} \psi_{t-1}(A_0, r_t, b_1^*, \ldots, b_p^*) = \min_{A_0}\{w_{t-1}(A_0) - d(r_t^k, A_0) + \chi_{t-1}(r_t, b_1^*, \ldots, b_p^*)\}$. If we take into account that χ_t is nondecreasing in t (because its terms are either constant distances or work function values that can only increase), we get that

$$\min_{A_0,b_0,\ldots,b_p} \psi_{t-1}(A_0, b_0, \ldots, b_p) \le \min_{A_0}\{w_{t-1}(A_0) - d(r_t^k, A_0)\} + \chi_t(r_t, b_1^*, \ldots, b_p^*),$$

and therefore

$$\min_{A_0,b_0,\ldots,b_p} \psi_t(A_0, b_0, \ldots, b_p) - \min_{A_0,b_0,\ldots,b_p} \psi_{t-1}(A_0, b_0, \ldots, b_p) \ge \min_{A_0}\{w_t(A_0) - d(r_t^k, A_0\} - \min_{A_0}\{w_{t-1}(A_0) - d(r_t^k, A_0)\}.$$

If we sum this over t, the left side telescopes and it is (up to an additive constant[3]) $\min_{A_0,b_0,\ldots,b_p} \psi_m(A_0, b_0, \ldots, b_p)$. Now, we are essentially done. The intuitive reason is that each value of work function in ψ_m is approximately equal to the optimal cost since they differ by the distance of two configurations (recall the property of work functions: $w_t(X) \le w_t(Y) + d(X, Y)$). Intuitively the value of ψ_m is approximately equal to the number ρ of work function values in it. More precisely, the value $\min_{A_0,b_0,\ldots,b_p} \psi_m(A_0, b_0, \ldots, b_p)$ is at most $\rho \min_X w_m(X)$ plus a constant, which shows that the Work Function Algorithm has competitive ratio at most $\rho - 1$.

We return to the example for $k = 2$ to illustrate the above approach. Consider again the expression $\psi_t(A_0, b_0, b_1, b_2) = w_t(A_0) - d(b_0^2, A_0) + w_t(\{b_0, b_1\}) + w_t(\{b_0, b_2\}) - d(b_1, b_2)$. Using the above proof outline, to show that the Work Function Algorithm is 2-competitive, it suffices to show that this expression is minimized when $b_0 = r_t$. Once we have the right ψ_t, this can be done by case analysis. We discuss only one case here: Let $A_0 = \{a_1, a_2\}$ and suppose that

$$\begin{aligned} w_t(\{a_1, a_2\}) &= w_{t-1}(\{a_1, r_t\}) + d(a_2, r_t) \\ w_t(\{b_0, b_1\}) &= w_{t-1}(\{r_t, b_1\}) + d(b_0, r_t) \\ w_t(\{b_0, b_2\}) &= w_{t-1}(\{b_0, r_t\}) + d(b_2, r_t). \end{aligned}$$

Then if we substitute these values, we observe that

$$\begin{aligned} \psi_t(A_0, b_0, b_1, b_2) = {} & \psi_{t-1}(\{r_t, b_1\}, r_t, a_1, b_0) \\ & + (d(a_2, r_t) + d(b_0, r_t) - d(b_0, a_2)) + (d(b_2, r_t) + d(r_t, b_1) - d(b_1, b_2)) \\ \ge {} & \psi_{t-1}(\{r_t, b_1\}, r_t, a_1, b_0). \end{aligned}$$

The other cases are handled similarly and we get a proof of the k-server conjecture for the special case of $k = 2$.

Theorem 2. *In every metric space, the Work Function Algorithm for the 2-server problem has competitive ratio* 2*.*

Why do we not just take $\psi_t = w_t(A_0) - d(r_t^k, A_0)$? Would this not prove a small competitive ratio? The short answer is that this ψ_t is not of the above form (even with $\chi_t = 0$). The above argument relies on the fact that r_t is not part of the definition of ψ_t; otherwise it should be replaced by r_{t-1} in ψ_{t-1} which would destroy the proof. After all, r_t is special for w_t because it is the last request, but it has no special meaning at all for w_{t-1}. The purpose of the expression χ_t is exactly this: to guarantee that $b_0 = r_t$ when we minimize ψ_t.

Can we find a good ψ_t expression for arbitrary k that proves the k-server conjecture? Unfortunately, we do not know how to do this, even for the case of $k = 3$. Perhaps this exact

[3]A minor technical detail: The constant is $\min_{A_0,b_0,\ldots,b_p} \psi_0(A_0, b_0, \ldots, b_p)$. For unbounded metric spaces, this may be unbounded in some cases. We have to be careful when selecting ψ to guarantee that this is not the case.

argument may not work for $k > 2$, but it is very possible that a similar approach works (for example, one in which χ_t contains also the min operator).

If we however are willing to settle for something less ambitious, the above argument works for every k (and it is in fact less involved than the case analysis of $k = 2$).

Theorem 3. *In every metric space, the Work Function Algorithm algorithm for the k-server problem has competitive ratio at most $2k - 1$.*

The proof is based on the following ψ_t:

$$\begin{aligned}\psi_t(A_0, b_0, A_1, b_1, \ldots, A_{k-1}, b_{k-1}) &= w_t(A_0) - d(b_0^k, A_0) + \sum_{i=1}^{k-1}(w_t(A_i) - d(b_i^k, A_i)) + k \cdot w_t(\{b_0, \ldots, b_{k-1}\}) \\ &= \sum_{i=0}^{k-1}(w_t(A_i) - d(b_i^k, A_i)) + k \cdot w_t(\{b_0, \ldots, b_{k-1}\}).\end{aligned}$$

This expression not only has the desired form but it is also symmetric. This makes it easier to apply the above reasoning: it suffices to show that when we minimize ψ_t *some* of the b_i's can be equal to r_t. But this is the least complicated part of the proof: Let $B = \{b_0, \ldots, b_{k-1}\}$ and let $w_t(B) = w_{t-1}(B - b_i + r_t) + d(b_i, r_t) = w_t(B - b_i + r_t) + d(b_i, r_t)$, for some $i = 0, \ldots, k-1$. Then, we can use the triangle inequality to substitute b_i by r_t in B. To simplify the presentation, let us focus on the part of ψ_t that involves b_i:

$$\begin{aligned}w_t(A_i) - d(b_i^k, A_i) + k \cdot w_t(B) &= w_t(A_i) - d(b_i^k, A_i) + k \cdot w_t(B - b_i + r_t) + k \cdot d(b_i, r_t) \\ &\geq w_t(A_i) - d(r_t^k, A_i) + k \cdot w_t(B - b_i + r_t).\end{aligned}$$

It follows that the minimum value of ψ_t is achieved when some of the b_i's is r_t. By the symmetry of the ψ_t expression, we can assume that $b_0 = r_t$. The expression ψ_t includes the sum of $2k$ values of w_t, and by the above argument, the Work Function Algorithm has competitive ratio $2k - 1$.

Can we improve the above? Many parts of the argument seem at first glance to be easy to improve, but there has been no progress in this direction in the last 15 years.

Similar arguments show that the Work Function Algorithm is k-competitive for special cases. The arguments differ slightly from the above, in the sense that the potential ψ_t may not have the above form exactly. However, the reasoning is very similar (and in most cases easier).

Metric spaces with $k + 1$ points. Take ψ_t to be the sum of the values of all possible configurations. There are $k + 1$ such configurations, and one of them is guaranteed to be a minimizer. Notice that we do not need the term $-d(b_0^k, A_0)$ or the duality property.

The 1-dimensional Euclidean space [BK04]. To simplify, consider the case where all requests are in some interval $[L, R]$. An easy argument shows that there is a minimizer which contains only copies of the extreme points L and R. Take $\psi_t = \sum_{i=0}^{k} w_t(L^i R^{k-i})$ and apply a similar argument as above.

Weighted star (or weighted paging) [BK04] A more complicated potential establishes the k-server conjecture for metric spaces where all the requests are placed at the leaves of a weighted star.

Metric spaces with $k + 2$ points [KP96]. For this case a non-trivial potential ψ_t is needed. Let $\mathcal{T}$ be the set of all spanning trees of the metric space M: Take $\psi_t = \min_{T\in\mathcal{T}} \sum_{[v_1,v_2]\in T}(w(M - v_1 - v_2) + d(v_1, v_2))$. It can be shown that ψ_t contains a minimizer. Since the tree has $k + 1$ edges, the sum inside ψ_t has $k + 1$ work function values. It follows with a similar argument as above that the Work Function Algorithm is k-competitive. A proof with a different potential was given in [BK04].

Extending this to metrics with $k + 3$ points does not seem to be easy.

For a different proof of the main result of this section see [Kou99].

4 Metrics with $k+1$ points

The special case of the k-server problem for metric spaces of $k+1$ points is of particular interest. It is much simpler than the general problem, but it is an important testing ground of many ideas.

For the deterministic case, the situation is much clearer than the general problem because given an online algorithm there is a unique optimal adversary: at any point in time, there is exactly one point of the metric not covered and this is where the adversary should place the next request. This creates a unique adversarial sequence of requests.

Also, by paying attention to the point not covered by the k servers, we get the equivalent evader problem: There is one evader which lives in the $k+1$ points of some metric space M. There is a sequence of requests and when the request is at the point covered by the evader the evader has to relocate. The objective is again to minimize the total distance traveled by the evader. In the context of randomized algorithms, the evader problem is also known as the cat and mouse game.

A natural algorithm for the k-server problem is the Balance Algorithm which tries to distribute the work fairly to the servers: For every server, the algorithm maintains the total distance traveled by the server so far; it services a request by the server that has the minimum sum of the distance traveled so far plus its distance to the request. In other words, the Balance Algorithm is the algorithm that greedily tries to be minmax fair (greedily minimizes the maximum accumulated distance among the servers). This algorithm raises game-theoretic issues to the k-server problem that have not been explored. Similar issues for other online problems have some outstanding open problems (and in particular the carpool fairness problem [AAN+98]).

Interestingly, the Balance Algorithm is k-competitive for metrics of $k+1$ points. In fact, if the Balance Algorithm is properly initialized, it becomes identical to the Work Function Algorithm for this special case. The reason behind it is the following: The defining property (7) of the Work Function Algorithm is $w_t(C_{t-1}) = w_t(C_t) + d(C_{t-1}, C_t)$. The correspondence with the Balance Algorithm is the following: the accumulated distance of a server at point x is equal to the work function at the remaining points. If the accumulated cost of the server that moves to the request r_t is $w_{t-1}(C_t) = w_t(C_t)$ before the move, it becomes $w_t(C_t)+d(C_{t-1}, C_t) = w_t(C_{t-1})$ after the move, which shows that the Balance Algorithm exactly parallels the behavior of the Work Function Algorithm.

The natural definition of Balance Algorithm applies also to metrics with more points, but not its competitiveness. In particular, the Balance Algorithm has unbounded competitive ratio even for the simple case of $k=2$ and 4 points. If we however make the Balance Algorithm more greedy by weighting differently the accumulated distance of the servers and the cost of moving to the request, the Balance Algorithm has bounded competitive ratio for $k=2$. For example, by weighting the distance to the request with a factor of 2, the competitive ratio is at most 10 [IR91] but no better than 6 [CL91a].

The study of randomized algorithms for $k+1$ points is much more complicated and interesting. We consider here memoryless randomized algorithms. Memoryless randomized algorithms are better described in the framework of the evader problem. A memoryless algorithm for the evader problem is defined by its probabilities p_{ij}: when the evader is at point i and there is a request at it, the evader relocates to point j with probability p_{ij}. The situation is captured essentially by the random walk on the points of the metric space with probabilities p_{ij}. The reason is the following: Without loss of generality, the online algorithm plays against a lazy offline algorithm. Suppose that the game starts at point a_0. The adversary places a request at a_0 and say that the offline algorithm moves to some point a_1. The online algorithm also moves randomly to some point other than a_0. The adversary knows only the probabilities but not the actual point of the online algorithm. The best strategy for the adversary is to request the points different from a_1 many times. This forces the online algorithm to take a random walk that ends at a_1. At the same time, the offline algorithm stays put at point a_1 incurring zero cost. After enough requests, the probability that the online evader converged to point a_1 is arbitrarily close to 1. Then the adversary places a request at a_1, the offline algorithm moves to some point a_2 and the situation is repeated. The cost of the offline algorithm is $\sum_{t=1}^{m} d(a_{t-1}, a_t)$, while the cost of the online algorithm is arbitrarily close to $\sum_{t=1}^{m} h(a_{t-1}, a_t)$, where $h(x, y)$ is the weighted hitting time of the random walk: the expected distance traveled by the random walk which starts at x and ends the first time it reaches y. The competitive ratio is (up to an additive term) the worst-case ratio of these costs. A simple observation is that worst-case happens when the points $a_0, \ldots, a_m$ move around a simple cycle (except perhaps an initial path that leads to this cycle). In summary, the competitive ratio of a memoryless randomized algorithm is equal to the maximum among all simple cycles c of the "stretch" of c: "stretch" is the expected distance of the random walk to visit all points of the cycle in order over the length of the cycle.

The following theorem shows that memorylessness is a severe restriction given that for metric spaces with $k + 1$ points we know that online algorithms with memory have polylogarithmic competitive ratio.

Theorem 4. *For every metric space of at least $k + 1$ points, no memoryless randomized algorithm has competitive ratio less than k.*

To see this, consider the stationary probabilities π_i of the random walk with probabilities p_{ij}; if the random walk has no stationary distribution, the situation can be handled similarly. Consider the inverse random walk that goes backwards in time with derivation probabilities $p'_{ij} = p_{ji}\pi_j/\pi_i$. This has the same stationary distribution $\pi'_i = \pi_i$. To get the lower bound, the adversary creates a random sequence following the inverse random walk. The claim is that on expectation each edge is traversed k times more by the online algorithm than the offline algorithm. The nice property of this argument is that the distances are essentially ignored; it is therefore sufficient to establish the claim for the well-studied unweighted random walks.

Intuitively, this lower bound is the analog of the deterministic lower bound. Consider again the situation of the deterministic lower bound where the online evader plays against k offline evaders and when the online evader moves from point a to point b, the offline evader at point b moves to point a. Now consider that the online algorithm does a random walk instead of moving deterministically and notice that the moves of each of the k adversaries follow the inverse random walk. Although the adversary against the randomized online algorithm does not know its position, with the help of the memorylessness property one can turn this intuition into a rigorous argument (for a more formal argument see [CDRS93]).

Interestingly, for every metric space of $k + 1$ points this lower bound is tight: there is a memoryless online algorithm with competitive ratio k. The description of an algorithm is much more intuitive for *resistive* metric spaces. Resistive metric spaces are defined as follows: Consider an electrical network of $k+1$ points consisting of resistances R_{ij} between every pair of points i and j. The effective resistance between every pair of points of such a network satisfies the triangle inequality, and it defines therefore a metric $M(R)$; such metrics are called resistive.

It follows from the theory of electrical networks that given a resistive metric space $M(R)$, the memoryless online algorithm with probabilities p_{ij} proportional to $1/R_{ij}$ is k-competitive. Given a metric $M(R)$, computing the resistances R can be done effectively (by essentially inverting a $k \times k$ matrix). However, for some matrices the computed resistances turn out to be negative. Thus not all matrices are resistive. However, there is a way to extend the above algorithm even for matrices that are not resistive as follows: For a non-resistive metric space M with distance function d, we find a resistive metric $M'(R')$ by contracting some edges The important property of $M'(R')$ is that when we consider the probabilities of the random walk the contracted edges have 0 probability. Therefore these edges are never traversed by the random walk and the analysis of the random walk on M and M' is identical, which shows:

Theorem 5. *For every metric space of $k+1$ points there is a memoryless randomized online algorithm with competitive ratio k.*

A more complete treatment of this subject can be found in [CDRS93]. The use of the theory of electrical networks has certain advantages and makes the algorithms and the analysis more intuitive; on the other hand, it sometimes obscures important issues and a lot of work is required to turn the intuition into rigorous proofs.

5 Memoryless randomized algorithms

In the previous section, we discussed randomized algorithms for the special case of metrics with $k+1$ points. We now turn our attention to randomized algorithms for arbitrary metric spaces. There are two types of randomized algorithms: memoryless algorithms and algorithms with memory. We limit our discussion to memoryless algorithms. There are important and deep results on randomized algorithms with memory which we do not discuss in this survey, but see Section 2 for pointers to the literature.

A natural randomized online algorithm is the Harmonic Algorithm which moves a server to the request with probability inversely proportional to the distance from it. The Harmonic Algorithm is a memoryless algorithm. This raises the question: Can memoryless algorithms be competitive? It is easy to see that every deterministic memoryless algorithm has unbounded competitive ratio (is it reasonable to be anything else than greedy in this setting?).

Theorem 6. *The Harmonic Algorithm has competitive ratio $O(2^k \log k)$. Furthermore, there are metric spaces for which the competitive ratio is at least $k(k+1)/2$.*

The proof of the lower bound is much simpler than the proof of the upper bound. It consists of analyzing the random walk of the Harmonic Algorithm on a metric space of $k+1$ points in which all distances are 1 with a single exception: a particular edge has distance ϵ (where ϵ is arbitrarily small).

It was shown by Raghavan and Snir [RS94] that for metric spaces of $k+1$ points the right answer in the theorem is exactly $k(k+1)/2$. More precisely, for these metric spaces the competitive ratio of the Harmonic Algorithm is at most $k(k+1)/2$ and there are metric spaces for which this bound is tight (within any δ). They also conjectured that this bound holds for arbitrary metric spaces.

Conjecture 3 (Harmonic Conjecture)**.** *In every metric space, the competitive ratio of the Harmonic Algorithm is at most $k(k+1)/2$.*

Raghavan and Snir expressed a stronger conjecture, the lazy adversary conjecture. To state it, we need to define the notion of lazy adversaries: a lazy adversary selects initially a submetric of $k+1$ points by adding a point to the original configuration; it places arbitrarily many requests on a configuration of k points of this subspace so that the online algorithm converges to this configuration with high probability. It then repeats the process starting from this configuration (to which it adds either a new point or the missing point of the original configuration). The lazy adversary conjecture states that in every metric space the worst-case adversary against the Harmonic Algorithm is a lazy adversary. Even if the lazy adversary conjecture holds, the result of [RS94] on metric spaces of $k+1$ points is not sufficient to prove the Harmonic Conjecture. The reason is that the lazy adversary can keep changing the submetric of $k+1$ points. However, Bartal, Chrobak, Noga, and Raghavan [BCNR02] extended the result in [RS94] and showed that the lazy adversary conjecture implies the Harmonic Conjecture.

The Harmonic Conjecture is true for the special case $k = 2$ [CL92a]. For $k = 3$, Berman, Karloff, and Tardos [BKT90] gave one of the earliest results on the Harmonic Algorithm, in which they showed that the competitive ratio is bounded (albeit extremely high). The best known upper bound (given by the Theorem 6) is due to Bartal and Grove. A slightly weaker result was shown initially by Grove [BG00]. The idea behind the upper bound is simple, but to make it work a lot of technical details have to be taken care of. The proof is based on a potential function argument. Let X be the configuration of the Harmonic Algorithm at some time and let Y be the configuration of the offline algorithm. We define the potential function $\Phi(X, Y)$ to express the disadvantage of the online configuration. The nice property of memoryless algorithms is that this potential depends only on the submetric spanned by the configurations X and Y and not on the history. The proof of Bartal and Grove uses a potential which is defined as follows: Let $f_1, \dots f_k$ be a decreasing sequence of numbers with $f_k = 1$. For a simple path $(a_1, \dots, a_s)$ define its scaled length to be $\sum_{i=1}^{s-1} f_i \cdot d(a_i, a_{i+1})$. Given a position x of an online server and a position y of an offline server, we define $\tilde{d}(x, y)$ to be the minimum scaled length among all simple paths $(x, a_2, \dots, a_{s-1}, y)$ where the intermediate points belong to the offline configuration Y. The potential $\Phi(X, Y)$ is taken to be $\tilde{d}(X, Y)$, i.e. the minimum weight of all complete matchings between X and Y with weights given by $\tilde{d}$. The most difficult part of the proof is to show that there are appropriate weights f_i, $i = 1, \dots, k$, so that when the Harmonic Algorithm moves, its expected cost is at most equal to the expected change in the potential Φ. For details see [BG00].

For the special case of $k = 2$, there are better memoryless algorithms than the Harmonic Algorithm. While the Harmonic Algorithm has competitive ratio 3, a simple variant of it has competitive ratio 2. The algorithm is simple: Let $\{x_1, x_2\}$ be the online configuration and r_t the next request. Then the online algorithm services r_t with the server at x_1 with probability $\frac{d(x_1,x_2)+d(x_2,r_t)-d(x_1,r_t)}{2d(x_1,x_2)}$; it uses the server at x_2 with the remaining probability. It is not hard to show that this algorithm is 2-competitive using the potential function $\Phi(X, Y) = 2d(X, Y) + d(x_1, x_2)$. This algorithm is identical to the algorithm of the resistive approach of the previous section for $k + 1$ points. It can be also interpreted as the randomized analog of the Double Coverage Algorithm (a fact which is also reflected in the potential).

6 Concluding remarks

The k-server problem is a fascinating subject with deep results and even deeper open problems. The two most important open problems are to settle the deterministic and the randomized k-server conjectures. Perhaps these problems together with the Splay Tree Conjecture are the most important open problems in online algorithms.

Another outstanding problem is to determine the competitive ratio of the Harmonic Algorithm. The special cases of the 3-server problem on general metrics and the k-server problem on the cycle, are important special cases that may lead the way to attack the k-server conjecture.

The k-server problem has been instrumental in the development of competitive analysis and it is likely that it will continue to influence the general area of decision-making under uncertainty.

6.1 Epilogue

This paper is part of a collection of surveys to honor **Christos Papadimitriou**, so it is fitting to add a short paragraph mentioning some of his contributions to the development of the theory of online algorithms. These represent only a small fraction of his work on a broad spectrum of research areas.

Besides his work on the k-server problem [KP94, KP95, KP96] which is explicitly mentioned here, Christos' work has been very important in the development of other aspects of online algorithms. He was a pioneer [PY89] on the subject of exploring and navigating in unknown environments and he formulated and studied important problems in this area [PY89, DP90, DKP91, KPY96]. His broad background in decision making and Economics are manifested in his work on the value of information and organizational theory [PY91a, DP92, Pap96] and on online aspects of linear programming [PY93]. He also proposed interesting refinements of competitive analysis such as the diffuse adversary and comparative analysis [KP00]. His treatment of online issues of congestion control [KKPS00] belongs to the large body of his work in the theory of networks. Finally, his conception of the price of anarchy [KP99] is part of his efforts to expand our understanding of decision-making under uncertainty and to import some aspects of competitive analysis to Game Theory.

References

[AAN+98] Miklós Ajtai, James Aspnes, Moni Naor, Yuval Rabani, Leonard J. Schulman, and Orli Waarts. Fairness in scheduling. *J. Algorithms*, 29(2):306–357, 1998.

[ACN00] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoret. Comput. Sci.*, 234:203–218, 2000.

[AKPW92] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its applications to the k-server problem. In Lyle A. McGeoch and Daniel D. Sleator, editors, *On-line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–10. AMS/ACM, 1992.

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. 37th Symp. Foundations of Computer Science (FOCS)*, pages 184–193. IEEE, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. 30th Symp. Theory of Computing (STOC)*, pages 161–168. ACM, 1998.

[BBBT97] Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proc. 29th Symp. Theory of Computing (STOC)*, pages 711–719. ACM, 1997.

[BBM01] Yair Bartal, Béla Bollobás, and Manor Mendel. A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In *Proc. 42st Symp. Foundations of Computer Science (FOCS)*, pages 396–405, 2001.

[BBN07] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. In *FOCS07*, pages 507–517, 2007.

[BCL02] Wolfgang Bein, Marek Chrobak, and Lawrence L. Larmore. The 3-server problem in the plane. *Theoret. Comput. Sci.*, 287:387–391, 2002.

[BCNR02] Yair Bartal, Marek Chrobak, John Noga, and Prabhakar Raghavan. More on random walks, electrical networks, and the harmonic k-server algorithm. *Inform. Process. Lett.*, 84:271–276, 2002.

[BDBK+90] Shai Ben-David, Allan Borodin, Richard M. Karp, Gabor Tardos, and Avi Widgerson. On the power of randomization in on-line algorithms. In *Proc. 22nd Symp. Theory of Computing (STOC)*, pages 379–386. ACM, 1990.

[BDBK$^+$94] Shai Ben-David, Allan Borodin, Richard M. Karp, Gabor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11:2–14, 1994.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BFKV92] Yair Bartal, Amos Fiat, Howard Karloff, and Rakesh Vohra. New algorithms for an ancient scheduling problem. In *Proc. 24th Symp. Theory of Computing (STOC)*, pages 51–58. ACM, 1992.

[BG00] Yair Bartal and Edward Grove. The harmonic k-server algorithm is competitive. *J. ACM*, 47(1):1–15, 2000.

[BK04] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theoret. Comput. Sci.*, 324:337–345, 2004.

[BKRS92] Avrim Blum, Howard Karloff, Yuval Rabani, and Michael Saks. A decomposition theorem and lower bounds for randomized server problems. In *Proc. 33rd Symp. Foundations of Computer Science (FOCS)*, pages 197–207. IEEE, 1992.

[BKT90] Piotr Berman, Howard Karloff, and Gabor Tardos. A competitive algorithm for three servers. In *Proc. 1st Symp. on Discrete Algorithms (SODA)*, pages 280–290. ACM/SIAM, 1990.

[BLMN03] Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric ramsey-type phenomena. In *STOC03*, pages 463–472, 2003.

[BLS87] Allan Borodin, Nathan Linial, and Michael Saks. An optimal online algorithm for metrical task systems. In *Proc. 19th Symp. Theory of Computing (STOC)*, pages 373–382. ACM, 1987.

[BLS92] Allan Borodin, Nathan Linial, and Michael Saks. An optimal online algorithm for metrical task system. *J. ACM*, 39:745–763, 1992.

[Bur94] William Burley. Toward an optimal online algorithm for layered graph traversal. Technical Report CS94-382, Department of Computer Science and Engineering, University of California at San Diego, 1994.

[BYCR88] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching with uncertainty. In *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 318 of *Lecture Notes in Comput. Sci.*, pages 176–189. Springer, 1988.

[CDRS93] Don Coppersmith, Peter G. Doyle, Prabhakar Raghavan, and Marc Snir. Random walks on weighted graphs and applications to on-line algorithms. *J. ACM*, 40:421–453, 1993.

[CKPV91] Marek Chrobak, Howard Karloff, Tom H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM J. Discrete Math.*, 4:172–181, 1991.

[CL91a] Marek Chrobak and Lawrence L. Larmore. On fast algorithms for two servers. *J. Algorithms*, 12:607–614, 1991.

[CL91b] Marek Chrobak and Lawrence L. Larmore. An optimal online algorithm for k servers on trees. *SIAM J. Comput.*, 20:144–148, 1991.

[CL92a] Marek Chrobak and Lawrence L. Larmore. HARMONIC is three-competitive for two servers. *Theoret. Comput. Sci.*, 98:339–346, 1992.

[CL92b] Marek Chrobak and Lawrence L. Larmore. Metrical service systems: Deterministic strategies. Technical Report UCR-CS-93-1, Department of Computer Science, University of California at Riverside, 1992.

[CL92c] Marek Chrobak and Lawrence L. Larmore. Metrical service systems: Randomized strategies. Manuscript, 1992.

[CL92d] Marek Chrobak and Lawrence L. Larmore. The server problem and on-line games. In Lyle A. McGeoch and Daniel D. Sleator, editors, *On-line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 11–64. AMS/ACM, 1992.

[CL94] Marek Chrobak and Lawrence L. Larmore. Generosity helps or an 11-competitive algorithm for three servers. *J. Algorithms*, 16:234–263, 1994.

[CMP08] Aaron Cote, Adam Meyerson, and Laura J. Poplawski. Randomized k-server on hierarchical binary trees. In *STOC08*, pages 227–234, 2008.

[CS04] Marek Chrobak and Jiří Sgall. The weighted 2-server problem. *Theoret. Comput. Sci.*, 324:289–312, 2004.

[DKP91] Xiaotie Deng, Tiko Kameda, and Christos Papadimitriou. How to learn an unknown environment. In *Proc. 32nd Symp. Foundations of Computer Science (FOCS)*, pages 298–303. IEEE, 1991.

[DP90] Xiaotie Deng and Christos Papadimitriou. Exploring an unknown graph. In *Proc. 31st Symp. Foundations of Computer Science (FOCS)*, pages 355–361. IEEE, 1990.

[DP92] Xiaotie Deng and Christos H. Papadimitriou. Competitive distributed decision making. In *International Federation for Infomation Processing, A-12*, pages 250–256, 1992.

[FFK+91] Amos Fiat, D. Foster, Howard Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. In *Proc. 32nd Symp. Foundations of Computer Science (FOCS)*, pages 288–297. IEEE, 1991.

[FKL+91] Amos Fiat, Richard Karp, Michael Luby, Lyle A. McGeoch, Daniel Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12:685–699, 1991.

[FM03] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, 2003.

[FRR90] Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms. In *Proc. 31st Symp. Foundations of Computer Science (FOCS)*, pages 454–463. IEEE, 1990.

[FRRS91] Amos Fiat, Yuval Rabani, Yiftach Ravid, and Baruch Schieber. A deterministic $o(k^3)$-competitive algorithm for the circle. Manuscript, 1991.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Systems Sci.*, 69(3):485–497, 2004.

[FW98] Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms: The State of the Art*. Springer, 1998.

[Gro91] Edward Grove. The harmonic k-server algorithm is competitive. In *Proc. 23rd Symp. Theory of Computing (STOC)*, pages 260–266. ACM, 1991.

[GW93] Gabor Galambos and Gerhard J. Woeginger. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM J. Comput.*, 22:349–355, 1993.

[IR91] Sandy Irani and R. Rubinfeld. A competitive 2-server algorithm. *Inform. Process. Lett.*, 39:85–91, 1991.

[Ira97] Sandy Irani. Page replacement with multi-size pages and applications to web caching. In *Proc. 29th Symp. Theory of Computing (STOC)*, pages 701–710. ACM, 1997.

[KKPS00] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Combinatorial optimization in congestion control. In *Proc. 41st Symp. Foundations of Computer Science (FOCS)*, pages 66–74, Redondo Beach, CA, 12–14 November 2000.

[KMMO94] Anna Karlin, Mark Manasse, Lyle McGeoch, and Susan Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11:542–571, 1994.

[KMRS86] Anna Karlin, Mark Manasse, Lyle Rudolph, and Daniel Sleator. Competitive snoopy caching. In *Proc. 27th Symp. Foundations of Computer Science (FOCS)*, pages 244–254, 1986.

[KMRS88] Anna Karlin, Mark Manasse, Larry Rudolph, and Daniel Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[Kou94] Elias Koutsoupias. *On-line algorithms and the k-server conjuncture*. PhD thesis, University of California, San Diego, CA, 1994.

[Kou99] Elias Koutsoupias. Weak adversaries for the k-server problem. In *Proc. 40th Symp. Foundations of Computer Science (FOCS)*, pages 444–449. IEEE, 1999.

[KP94] Elias Koutsoupias and Christos Papadimitriou. On the k-server conjecture. In *Proc. 26th Symp. Theory of Computing (STOC)*, pages 507–511. ACM, 1994.

[KP95] Elias Koutsoupias and Christos Papadimitriou. On the k-server conjecture. *J. ACM*, 42:971–983, 1995.

[KP96] Elias Koutsoupias and Christos Papadimitriou. The 2-evader problem. *Inform. Process. Lett.*, 57:249–252, 1996.

[KP99] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proc. 16th Symp. on Theoretical Aspects of Computer Science (STACS)*, Lecture Notes in Comput. Sci., pages 404–413. Springer, 1999.

[KP00] Elias Koutsoupias and Christos Papadimitriou. Beyond competitive analysis. *SIAM J. Comput.*, 30:300–317, 2000.

[KPY96] E. Koutsoupias, C. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. 23rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 280–289, Paderborn, Germany, 8–12 July 1996.

[KT00] Elias Koutsoupias and David Taylor. The CNN problem and other k-server variants. In *Proc. 17th Symp. on Theoretical Aspects of Computer Science (STACS)*, volume 1770 of *Lecture Notes in Comput. Sci.*, pages 581–592. Springer, 2000.

[LO06] Lawrence L. Larmore and James A. Oravec. T-theory applications to online algorithms for the server problem. *CoRR*, abs/cs/0611088, 2006.

[MMS88] Mark Manasse, Lyle A. McGeoch, and Daniel Sleator. Competitive algorithms for online problems. In *Proc. 20th Symp. Theory of Computing (STOC)*, pages 322–333. ACM, 1988.

[Pap96] Christos H. Papadimitriou. Computational aspects of organization theory. In *ESA96*, pages 559–564, 1996.

[PY89] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. In *Proc. 16th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 372 of *Lecture Notes in Comput. Sci.*, pages 610–620. Springer, 1989.

[PY91a] Christos H. Papadimitriou and Mihalis Yannakakis. On the value of information in distributed decision making. In *Proc. 10th Symp. on Principles of Distributed Computing (PODC)*, pages 61–64. ACM, 1991.

[PY91b] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoret. Comput. Sci.*, 84:127–150, 1991.

[PY93] Christos H. Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In *Proc. 25th Symp. Theory of Computing (STOC)*, pages 121–129. ACM, 1993.

[RS89] Prabhakar Raghavan and Marc Snir. Memory versus randomization in online algorithms. In *Proc. 16th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 372 of *Lecture Notes in Comput. Sci.*, pages 687–703. Springer, 1989.

[RS94] Prabhakar Raghavan and Marc Snir. Memory versus randomization in on-line algorithms. *IBM J. Res. Dev.*, 38:683–707, 1994.

[SSdP03] René Sitters, Leen Stougie, and Willem E. de Paepe. A competitive algorithm for the general 2-server problem. In *Proc. 30th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 2719 of *Lecture Notes in Comput. Sci.*, pages 624–636. Springer, 2003.

[ST85a] Daniel Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208, 1985.

[ST85b] Daniel D. Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32:652–686, 1985.