The K-server Problem

Lam T. Nguyen

Danika Passler Bates

December 28th, 2020

Abstract:

The k-server problem has been at the heart of online algorithm research for quite some time now. It has useful applications across different fields for the problem encompasses and is encompassed in other online problems. Multiple discoveries were made in the late 90s, and we can still see other new variations to this day. A particular element surrounding the problem that captures special attention is the efficiency of adversary models and the optimal solution to determine algorithms' performance. In the following paper, we aim to establish the most recent report on the research related to the k-server problem, with special coverage on adversarial models to demonstrate a bigger picture.

1 Introduction

K-server is a prominent problem in the study of online algorithms. The problem itself has simple principles such that many problems in the world can be reduced to k-server. The k-server problem was originally presented in 1988 as a means of generality for the popular page replacement algorithms during the 60s and 70s [1]. Some of the other practical instances of the k-server problem include paging, weighted paging, two-headed disk, k-taxicab, the chasing problem, and many more [1]. Another important application of the k-server problem is distributed systems. Nishida et al [2] described models and algorithms to distribute client and server matching efficiently based on load balancing, a similar instance to the weighted k-server problem.

Problem Definition: There are three simple variables: A metric space M, some k servers, and an input sequence σ. The metric M can be weighted or unweighted. There is no bound to how many metric points there are. Input sequence σ requests a particular point on the metric. The algorithm responds by moving a server towards the demanded point, should there not be any server at the requested location. It is assumed that all servers are available to serve the active request, and that requests cannot be cancelled. The cost of the algorithm is the distance to move each server..The problem's objective is therefore to minimize the total distance travelled by the servers [3].

The k-server qualifies as an online problem.Online problems differ from offline problems in that in an online problem, the input is not known ahead of time and decisions must be made as new inputs are revealed. Meanwhile, an offline algorithm sees the entirety of the sequence and consequently, can make clever decisions. Online problems can be more practical than offline problems and pose a more challenging problem for researchers. For the online k-server problem, request points are given one by one at a time. A seemingly efficient choice at a given time may not lead to the best choice for the whole sequence and vice versa. Inevitably, achieving an optimal solution is difficult because a decision might be made that isn't ideal given the future input.

For online problems, the metric that is used most often to judge the effectiveness of an algorithm is not the time complexity as is often analyzed in offline problems, but is the competitive ratio. The competitive ratio is the ratio between the cost of an online algorithm, and the cost of an optimal offline solution which knows the entire input. When examining competitive ratios, we tend to look at worst case inputs. To conduct the worst case analysis, we imagine that an adversary is purposefully impairing our algorithm by feeding wasteful inputs. This adversary can have varying abilities, depending on the adversarial model used. Given that the model in use has an effect on the performance of an algorithm, we encourage readers to familiarize themselves with the powers that adversaries can have to gain a deeper understanding of the analysis of the k-server and other problems. In this paper, we survey the k-server problem, its algorithms, and the various adversarial models that are used in order to analyze these algorithms. In particular, we study the following adversaries:

- Oblivious Adversary: An adversary who knows the general strategy of the algorithm, but does not know about any random bits if the algorithm uses randomization
- Adaptive Online Adversary: An adversary that makes the next request based on the algorithm's previous answers. The adversary serves the next request immediately [4].
- Adaptive Offline Adversary: An adversary that makes the next request based on the algorithm's previous answers, and serves the requests optimally at the end [4]. This adversary is so strong that randomization does not help against it [4], so a randomized algorithm against an Adaptive Offline Adversary is essentially equivalent to a deterministic algorithm.
- Weak Adversaries: A weak adversary is one in which the optimal offline algorithm has less servers than the algorithm

The type of metric space used for the k-server problem can have a great impact on the analysis of algorithms. There are several spaces that have been studied for the k-server problem, including:

- General metric spaces: This is the general case of the k-server problem described above
- Uniform metric spaces: Distance between every pair of points is the same [5]
- Asymmetric metric spaces: Spaces where the metric is a weighted, directed graph [6]
- Resistive metric spaces: These metric spaces represent an electrical network, and therefore have a resistance between each pair of vertices in addition to a distance. A metric with node set V, and cost matrix d is considered to be resistive if its' resistive inverse exists: a metric with node set V and branch resistances d' such that for all $u \neq v$ the effective resistance R_{uv} between u and v is d_{uv} [6].
- Line metric spaces
- Trees

2 Big Picture

2.1 Existing Algorithms

The k-server problem was introduced in 1988 in Manasse et al.'s "Competitive Algorithms for Server Problems" [1] as a generalization to model the paging and caching problems. Therefore, it suits our purpose here to brief these related problems. The paging problem involves two memory blocks of different sizes, and they store pages. Requests are made on the smaller block. Many similarities to the k-server problem can be seen here. One can gather all the request locations of the k-server together as a small block. The current configuration of servers acts as the big block in turn, and server traveling distances are uniform in the paging problem where pages are evicted and brought in simultaneously.

The caching problem is another variant of the paging problem on asymmetric spaces.

For the ease of understanding, we will denote *k* as the number of servers, and *n* as the length of input sequence.

For the paging problem, an optimal offline algorithm has access to future requests and evaluates the best serving sequence. Belady [7] coined it as the optimal replacement scheme that favors the longest forward distance called MIN. However, assessing online algorithms by the usual worst-case analysis does not do any good for Irani [8] argues that the nature of paging is exploitable by requesting the most recent evicted page for a guaranteed fault. Hence, for the deterministic setting, no paging algorithm can achieve a competitive ratio better than k for a small memory block of size k. Standard deterministic paging algorithms are First-In-First-Out (FIFO), Least-Recently-Used (LRU), and Flash when full (FWF). A group of paging algorithms are categorized as being part of the Marking class, a randomized paging strategy that marks hit pages when delivered in phases. As for their performance, Marking algorithms have a competitive ratio of $2H_k$ against an oblivious adversary that chooses a lazy algorithm as uncovered by Fiat et al. [9], H_k being the k'th harmonic number. Lykouris et al. [10] modified the marking class and incorporated machine learning prediction to arrive at the Predictive Marker algorithm which is bounded by O(log*k*). Most recently, Jiang et al. [11] studied the weighted paging problem with predictions under the strong per request prediction model and gave a 2-competitive algorithm.

There are greedy online algorithms for the k-server problem as well. By its nature, it chooses the choice that maximizes the cost in the short term by moving the closest server. The competitive ratio for the algorithm is not competitive. Borodin et al. [6] demonstrated that in a two servers setting, one server can be used to serve all the requests, which gives a boundless ratio. Nevertheless, it serves well as the base algorithm for others to build upon. The universal Work Function algorithm was based upon the principle of Retrospective Greedy. Instead of minimizing the cost as the distance traveled, the Work Function Algorithm minimizes the total transitions between configurations or "workload." It achieves a competitive ratio of 2k - 1 for general metrics [12]. This discovery has such influence that it is taken up frequently in later research. Mentionable works such as [13] tested Work Function in network flows to evaluate the practical applicability against those non-competitive to less competitive algorithms, namely Greedy or Balance. Balance is a lazy algorithm that evens the distance traveled across all servers upon requests. It is believed that Balance is a generalization of First-In-First-Out principles [14]. Initially proven to be (n - 1)-competitive for symmetric (n - 1)-servers problem [1], Balance sparked the interest for cache problems, especially weighted instances [15]. Bogdan Chlebus from the University of California worked alongside Chrobak and self-sufficiently proved Balance to be k-competitive for any metric space with n = k + 1 points [6]. Efforts were put into study metric spaces where there are less than n - 1 servers, and an exciting instance exists that is 2-competitive for two servers [15]. The concept of comparing server residues formed the algorithm RES.

The k-server conjecture is an important open question for the k-server problem that proposes that there is a deterministic algorithm for any metric space and any number of servers with a competitive ratio of *k*. Chrobak et al. [15] provided a satisfying algorithm for tree and line metrics. The algorithm is called Double Coverage Algorithm (DCA). Elements of DCA rely upon greedy movements on a line, which moves servers on either one or both sides of the request point. Around that time, a *k*-competitive algorithm ROTATE was found for the metric space for unweighted-cache problems. Essentially, it moves the furthest server a distance picked between 0 and 1 arbitrarily for a uniform metric [15].

The resource augmentation setting, or the (h,k)-server problem is a model where the optimal offline algorithm possesses less servers than the algorithm. The number of servers that the optimal algorithm possesses is denoted by the variable h. Young [14] recognized Least-Recently-Used and Balance to be mirror perspective algorithms in the duality principle of optimization. He introduced Greedy Dual, a deterministic primal-dual algorithm that generalizes both LRU and Balance and maintains a competitive ratio less than *k* for the resource augmentation setting or the (h,k)-server problem, of $\frac{k}{k-h+1}$. This was not possible without Sleator et al.'s [16]'s original bound of $\frac{k}{k-h+1}$ within pages of uniform size. The variable *h*, the optimal solution's servers, was the MIN algorithm's servers in Sleator's work while *k* is any online algorithms' servers. While $\frac{k}{k-h+1}$ is the best bound known for the (h,k)-server problem, the (h,k)-server conjecture that such bound holds for any metric space is not true. Borodin et al.'s [6] has proven that RES algorithm's 2-competitiveness for two servers is the best bound for when the optimal solution has two servers, so it rejects the (h,k)-server conjecture.

The idea of having an adversary to evaluate algorithms against adapts from the zero-sum game theory which involves the outcomes of two participants affecting each other. Minimax is deemed to help with worst-case analysis. Because the other player of the game can create obstacles based on their knowledge of our algorithm, randomized choices can help that confuse the adversary and achieve a better bound while deterministic cases run into dead ends [18]. Just as randomized marking algorithms achieve better results than deterministic paging algorithms, randomization can help with the k-server problem as well.

A circle can be viewed as a line when the CIRC algorithm places a non-passable point at random. An oblivious adversary would only result in a 2k-competitive ratio if the metric space presented can be embedded into a circle [6]. In another instance, in a space where there are k+1 points, the CIRC2 algorithm is an improvement based on dividing a circle into arcs. It applies a k-competitive algorithm to the requested segments. This yields a competitive ratio of O($\sqrt{NlogN}$) against oblivious adversaries, where N is the number of even-spaced points on the circle [19]. Similarly to the deterministic k-server conjecture, a randomized k-server conjecture operates the same way, proposing that there is a bound of O(log*k*) [18]. In light of the deterministic approach for circles, Fiat et al. [20] also applied Double Coverage on segments with exhaustion marking. The algorithm works in phases and subphases. O(k^3) competitive ratio is attained by showing 2k subphases of 4k^2 optimal cost for every phase. Subphases' cost adapts from Ben-David et al [4]'s corollary, which states there is a deterministic 4k^2-competitive algorithm for a circle for a total server count less than the number of arbitrary nodes placed on a circle. This was built upon another corollary that states the existence of a randomized algorithm against an adaptive online adversary implies another existence of deterministic algorithm with randomized ratio squared-competitiveness against adaptive offline adversaries.

Nikhil Bansal has made astonishing breakthroughs on randomization lately. His work during 2011 was the first polylogarithmic-competitive algorithm on hierarchically well-separated trees (HST), with a significant O($\log^2 k \log^3 n$) competitiveness [21]. HSTs result from embedding generic spaces into structured trees. This was done to provide a probabilistic distribution for randomization purposes. Bansal's 2011 paper laid the groundwork for Bansal's Fractional Allocation algorithm. Coté et al. offered an expected O($\log^2 \triangle \log n$) ratio for general metrics' diameter $\triangle$ when embedded into high degree trees [22] . A polylogarithmic randomized competitive ratio was unseen until 2015. That was when Bansal et al. [3] managed to acquire an Õ($\log^3 n \log^2 k$) bound for randomized algorithm on any n point spaces. Their work involves HSTs in a weighted context, along with suggestions of how the allocation problem translates to k-server.

Randomization on a line metric has been studied as well. Csaba et al. designed an O($n^{2/3}$ logn)-competitive evader algorithm against an oblivious adversary [23]. An l-evader problem is an instance of k-server where each point in a metric can provide residence for more than one evader [24]. In contrast, the generic k-server can only have one server at one point at a time.

Laszlo mentioned a random walk on a graph as a finite stochastic model that places the current state's importance on the last event's probability on a weighted directed graph [25]. It has a tight relationship with the electrical network in graph theory, especially the resistive inverse - a value that constitutes an optimal stretch of a random walk. Coppersmith et al. suggest that such a technique can help design a randomized strategy named RWALK for resistive metric spaces and k + 1 vertices metric spaces [26]. They delivered a k-competitive randomized algorithm against oblivious adversaries and 2n-1 lower bound for randomized algorithms against adaptive adversaries.

2.2 Bounds

	Best Lower bound	Best Upper bound
Deterministic	k	$2k$-1
Randomized-Oblivious Adversary	$\frac{logk}{log^2 logk}$	$2k$-1
Randomized-Adaptive Adversary	k	$2k$-1
Weak Adversary	$\frac{k}{k-h+1}$	$2h$*opt(h)-opt(k)+O(1)

The offline version of the k-server problem is best optimized to O(kn^2) where n is the number of requests in a sequence. This is done by reducing it to a minimum cost / maximum flow problem [6]. For the online version, the deterministic lower bound is found to be no less than k, the number of servers in a configuration [4]. For the general metric space, there is an upper bound of O(2k-1) with the work function algorithm [12].

On the other hand, randomization allows the online player more power and hence better results. Fiat et al discovered an upper bound for the competitive ratio of O(logK) for a randomized algorithm for the paging problem [4]. This is equivalent to the k-server problem in a uniform metric space. The randomized k-server conjecture proposes that this bound holds for any metric space. [18]. No upper bound better than the O(2k-1) bound by Katsoupias et al. has been found for randomized algorithms in general metric spaces. [27]

A lower bound of Ω(log K/ log^2 logK) against an oblivious adversary for any metric space was found by Bartal et al [27]. Previously, a lower bound of Ω(log log k) was found for any metric space with at least k+1 points [28]. When faced with a stronger adversary, randomization falls short. The randomized lower bound against an adaptive adversary is k [28] , which is no better than the deterministic case.

For weak adversaries, as in the (h,k)-server problem, the lower bound for deterministic algorithms is $\frac{k}{k-h+1}$ for uniform metric spaces and for metric spaces with at least k+1 points[Bansal et al, 2016]. Katsoupias found that the WFA is 2h-competitive with an upper bound of $2h \cdot \mathrm{OPT}_h - \mathrm{OPT}_k + O(1)$ for the (h,k)-server problem [29]. The subscripts in this bound represent the number of servers used by the algorithm.

2.3 Advice

In online algorithms design, we often study how much advice an algorithm would need to know about the future input to achieve an optimal solution or how advice about the information can improve the competitive ratio. The advice model is helpful in mediating between the algorithm and cruelly created inputs to tax the algorithm to its worst possible case. Some can see that would only limit online algorithm research to improving the worst case, while the support of advice bit was another untapped research field that many can explore. At first, the concept of advice was brought in by Emek et al. [30].They argued that algorithms are handicapped by a lack of sequence premonition and thus unable to produce fair results. They identified the range [$\Theta(1)$, $\lceil \log k \rceil$] is suitable to provide b bits of advice per request that can create a $k^{O(1/b)}$-competitive deterministic algorithm under general metric spaces. This early advice model had its bits following any single request, while later models had their advice bits on an advice tape. The advice tape contains the knowledge of the input. However, our algorithm can only digest certain bits of advice while taking individual information. This steers the advice research into optimizing the amount of advice given at each time. In the generalization of the k-server problem where all distances are uniform, i.e., the paging problem, linear advice is needed for an optimal solution [31]. Böckenhauer et al. examined the k-server problem with advice under simple metric spaces, namely metric graphs with vertex set V and edge set E. They found at least n/2(log$k$ - c) advice bits, for some constant c < 1.443, where the number of requests n is a multiple of 4k bits was needed [32]. This improvement was short-lived because a year later Renault and Rosén gave a $\frac{logk}{b-2}$-competitive online algorithm for general metric spaces with b bits of advice per request, where $3 \le b \le \log k$ [33]. Gupta et al. explored the k-server problem with advice under various sparse graphs such as paths, treewidth α space, and collective tree spanners [34]. They discovered that Θ(n) bits is sufficient to accomplish a 1-competitive algorithm for paths. Furthermore, an α-width tree needs an amount of at least $\frac{n}{2}$(logα - 1.22) bits for optimality, but α has to be in the bounds of [4, 2k]. Remarkably, an almost linear amount of advice, about O(n(logα + loglog$N$)) can assist optimal performance in a constant-width tree.

3 Harmonic Algorithm

The survey would not be complete without a discussion on the significance of our favorite k-server algorithm: Harmonic. A short history on the background of Harmonic takes us back to Raghavan et al.'s 1989 paper that suggested a randomized online algorithm for the cache problem [35]. Harmonic adapts elements from printers' fonts cache where operations cost differently. Resources directed to serve in harmonic depend on the inverse probability of servers' respective location to a specified request slot. Any close-by request should have a smaller distance and, thus, more likely to be moved, but the action is done randomly to avoid greediness. They proved that Harmonic is $\frac{k(k+1)}{2}$ competitive to any lazy adversaries when introducing the Lazy Adversary Conjecture. It should be noted that this conjecture has been proven to be a failure by Peserico [36]. This is due to adjustments in laziness that can generate noncompetitive outcomes.

Nonetheless, the lazy adversary inspired the randomized Harmonic algorithm that we are examining and another adapted version that is the best randomization bound known, the RWALK algorithm [26]. Berman et al. had an early attempt that got harmonic 3-competitive with three servers, at a tremendous cost of $3^{17,000}$ against online adaptive adversaries[37]. Grove had his eyes on the problem and presented an O($\frac{5}{4}k * 2^{k} - 2k$)-competitive ratio for Harmonic [38] and carried on his work until Bartal et al. showed Harmonic to be O($2^{k} * logk$)-competitive [39].

Harmonic is defined as follows. Let a normalizing constant N have a formula of:

$$N = \sum_{y} 1/d(X, y)$$

for requested points X and y as positions of our servers. Depending on their distance, a server located at point z can be chosen with a probability of $\frac{1}{N^*d(X,z)}$). Ben-David et al. provided researchers with a framework to study online algorithms by involving them in request-answer games [4]. The adversary can present our algorithm with requests and decide the initial configuration for the problem. The adaptive offline adversary has rendered randomization techniques trivial. Therefore, the adaptive online adversary is the focus of algorithm analysis. Besides, given the adaptive ability, it can calculate the cost of requests in real-time to adjust inputs. For the Harmonic algorithm, its adversary is defined to have an equal number of servers as the algorithm on the metric space. It can see the randomness from earlier stages and how the configuration changed thus far. It can also control the length of the input sequences. The adversary and our algorithm will, in turn, respond to requests using the same set of servers. Our adversary takes odd phases while harmonic serves on even ones. There is no guarantee the adversary's server move should occur, but a request will be generated wherever its chosen server's location is after the turn.

Bipartite servers matching and potential functions are concepts vital to the correctness of Harmonic. These factors serve to show that the algorithm is competitive. The current metric space is a weighted bipartite graph so that by server matching, we seek to optimize matching based on the least weight possible when the algorithm acts. Analysis's best interest is for the adversary to do the least work possible so that Harmonic has to serve all the requests. Grove wanted to demonstrate the cost gap between Harmonic and the adversary does not fluctuate significantly throughout the sequence. The potential function is labeled as the upper bound of work done by the algorithm that takes the server's position as input. This echoes back to the Work Function analysis, provided probability factors play a role in decision making. The potential function has to obey three following conditions in order to be competitive, which is a lemma proven afterward by Bartal et al. [39]:

- It has to be non-negative
- For any phases after the first phase and a phase before it, their difference over distance moved is bounded by some C(*k*)
- Expected difference of potential value between a mid and end of a phase is bounded by negative the expectation over coin tosses in such phase for distance moved by the algorithm in that phase

Expectation over coin tosses is defined as $\sum_{i=1}^{\infty}\left(\frac{1}{2}\right)^{i}$ for the *i*th phase. A non-increasing monotonic function calculates the weight of steps on a path. Lemma's proof involves mathematical induction. It proves that for each request in the sequence, the Harmonic cost is always approximately C(*k*) times the adversary cost plus additive advantage constant. However, the potential function has to be proven to meet the predefined constraints. Bartal investigated weight fluctuation caused by minimum weight matching between two points and derived at bounds for both cases when weight matching for both points intersects and does not. After revisions against three properties, Harmonic is C(*k*)-competitive as a result. However, for the third condition to suffice, the first two conditions are embodied to bound the expected value of algorithm cost, reduced to a recurring bound primarily relying on previous monotonic function. Verifying the monotonicity of the function gives Harmonic a $((k+1)(2^k - 1) - k)$-competitiveness. Bartal reiterates the potential function with an added adversary server on a server's path to matching, hopefully improving the upper bound from the last time. Small adjustments helped to keep the new function in line with conditions before being competitive. Especially for condition three, they are reanalyzing potential function changes with a revised non-increasing function allowed for a slightly better bound of O(2^klog$k$)-competitiveness. Overall, the Harmonic algorithm, over a long period, is a continuous effort from the basis of Harmonic sequence to an enormous ratio that finally made it to 2^klog$k$ from the previous ((k + 1)(2^k - 1) - k) which was only $\frac{5}{4}k \ * 2^k - 2k$ nine years ago. Harmonic itself shows to have lent elements from various other algorithms that came before it. It achieved a significant bound against an adaptive opponent in a randomization setting, as thorough constant reaffirmation and re-examination have shown across the analysis.

4 Concluding Remarks

In this paper, we have presented a survey of the k-server problem. We have given a big picture of the various algorithms that have been developed for the problem, and have surveyed how different adversarial models affect the analysis of the algorithms. We paid particular attention to the harmonic algorithm because of its performance and study under various adversarial models.

There remains much research to be done on the k-server problem. The randomized k-server conjecture that there exists a randomized algorithm with competitive ratio log k for any metric space is also open. The k-server conjecture that there exists a deterministic algorithm with competitive ratio k for any metric space also remains open, and is the most important open problem with respect to the k-server problem. As the k-server conjecture has been open since Manasse et al proposed it in 1988 [1], it would be extremely significant if it were to be proven.

References:

[1]: M. Manasse, L. Mcgeoch, and D. Sleator, “Competitive algorithms for on-line problems,” *Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC '88*, 1988.

[2]: H. Nishida and T. Nguyen, “Optimal Client-Server Assignment for Internet Distributed Systems,” *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011.

[3]: N. Bansal, N. Buchbinder, A. Madry, and J. (S. Naor, “A Polylogarithmic-Competitive Algorithm for the k-Server Problem,” Journal of the ACM, vol. 62, no. 5, pp. 1–49, 2015.

[4]: S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson, “On the power of randomization in online algorithms," Proceedings of the twenty-second annual ACM symposium on Theory of computing - STOC ’90, 1990.

[5]: N. Bansa, M. Eliáš, G. Koumoutsos, and J. Nederlof, “Competitive Algorithms for Generalized k-Server in Uniform Metrics,” *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 992–1001, 2018.

[6]: A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge, U.K.: Cambridge University Press, 1998.

[7]: L.A. Belady. A study of replacement algorithms for virtual storage computers. IBM Systems Journal, 5:78-101, 1966.

[8]: S. Irani. Competitive analysis of paging: A survey, In Proc. of the Dagstuhl Seminar on Online Algorithms, Dagstuhl, Germany, June 1996.

[9]: A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator and N. E. Young, Competitive Paging Algorithms, 1988.

[10]: T. Lykouris and S. Vassilvtiskii, “Competitive caching with machine learned advice,” *International Conference on Machine Learning*, pp. 3302–3311, 2018.

[11]: Z. Jiang, D. Panigrahi, and K. Sun, “Online Algorithms for Weighted Paging with Predictions,” *arXiv.org*, 16-Jun-2020. [Online]. Available: https://arxiv.org/abs/2006.09509. [Accessed: 28-Dec-2020].

[12]: E. Koutsoupias and C. H. Papadimitriou, “On the k-server conjecture,” *Journal of the ACM*, vol. 42, no. 5, pp. 971–983, 1995.

[13]: T. Rudec, A. Baumgartner, and R. Manger, “A fast work function algorithm for solving the k-server problem,” *Central European Journal of Operations Research*, vol. 21, no. 1, pp. 187–205, 2011.

[14]: N. Young, “The k-server dual and loose competitiveness for paging,” *Algorithmica*, vol. 11, no. 6, pp. 525–541, 1994.

[15]: M. Chrobak, H. Karloof, T. Payne, and S. Vishwnathan, “New Results on Server Problems,” *SIAM Journal on Discrete Mathematics*, vol. 4, no. 2, pp. 172–181, 1991.

[16]: D. D. Sleator and R. E. Tarjan, “Amortized efficiency of list update and paging rules,” *Communications of the ACM*, vol. 28, no. 2, pp. 202–208, 1985.

[17]: N. Bansal, M. Eliéš, Ł. Jeż, and G. Koumoutsos, “The (h,k)-Server Problem on Bounded Depth Trees,” *ACM Transactions on Algorithms*, vol. 15, no. 2, pp. 1–26, 2019.

[18]: E. Koutsoupias, “The -server problem,” *Computer Science Review*, vol. 3, no. 2, pp. 105–118, 2009.

[19]: A.Blum, P.Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. SIAM Journal on Computing, 26:110-137, 1997.

[20]: A. Fiat, Y. Rabani, Y. Ravid, and B. Schieber, “A deterministicO(k^3)-competitive k-server algorithm for the circle,” *Algorithmica*, vol. 11, no. 6, pp. 572–578, 1994.

[21]: N. Bansal, N. Buchbinder, A. Madry, and J. Naor, “A Polylogarithmic-Competitive Algorithm for the k-Server Problem,” *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 2011.

[22]: A. Coté, A. Meyerson, and L. Poplawski, “Randomized k-server on hierarchical binary trees,” *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*, 2008.

[23]: B. Csaba and S. Lodha, “A randomized on–line algorithm for the k–server problem on a line,” *Random Structures and Algorithms*, vol. 29, no. 1, pp. 82–104, 2006.

[24]: E. Koutsoupias and C. Papadimitriou, “The 2-evader problem,” *Information Processing Letters*, vol. 57, no. 5, pp. 249–252, 1996.

[25]: L. Lovász, “Random Walks on Graphs: A Survey, Combinatorics, Paul Erdos is Eighty,” *Bolyai Soc. Math. Stud.*, vol. 2, Jan. 1993.

[26]: D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir, “Random walks on weighted graphs and applications to on-line algorithms,” *Journal of the ACM*, vol. 40, no. 3, pp. 421–453, 1993.

[27]: Y. Bartal, B. Bollobas, and M. Mendel, “A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems,” *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[28]: H. Karloff, Y. Rabani, and Y. Ravid, “Lower Bounds for Randomized k-Server and Motion-Planning Algorithms,” *SIAM Journal on Computing*, vol. 23, no. 2, pp. 293–312, 1994.

[29]: N. Bansal, M. Eliáš, Ł. Jeż, G. Koumoutsos, and K. Pruhs, “Tight Bounds for Double Coverage Against Weak Adversaries,” *Theory of Computing Systems*, vol. 62, no. 2, pp. 349–365, 2016.

[30]: Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén, “Online computation with advice,” *Theoretical Computer Science*, vol. 412, no. 24, pp. 2642–2656, 2011.

[31]: J. Boyar, L. M. Favrholdt, C. Kudahl, K. S. Larsen, and J. W. Mikkelsen, “Online Algorithms with Advice,” *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–34, 2017.

[32]: H.-J. Böckenhauer, D. Komm, R. Královič, and R. Královič, “On the Advice Complexity of the k-Server Problem,” *Automata, Languages and Programming Lecture Notes in Computer Science*, pp. 207–218, 2011.

[33]: M. P. Renault and A. Rosén, “On Online Algorithms with Advice for the k-Server Problem,” *Approximation and Online Algorithms Lecture Notes in Computer Science*, pp. 198–210, 2012.

[34]: S. Gupta, S. Kamali, and A. López-Ortiz, “On Advice Complexity of the k-server Problem under Sparse Metrics,” *Structural Information and Communication Complexity Lecture Notes in Computer Science*, pp. 55–67, 2013.

[35]: P. Raghavan and M. Snir, “Memory versus randomization in on-line algorithms,” *Automata, Languages and Programming Lecture Notes in Computer Science*, pp. 687–703, 1989.

[36]: E. Peserico, “The Lazy Adversary Conjecture Fails,” *Theory of Computing Systems*, vol. 37, no. 3, pp. 397–403, 2004.

[37]: P. Berman, H. Karloff, and G. Tardos, “A competitive 3-server algorithm,” *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pp. 280–290, Jan. 1990.

[38]: E. F. Grove, “The harmonic online K-server algorithm is competitive,” *Proceedings of the twenty-third annual ACM symposium on Theory of computing - STOC '91*, 1991.

[39]: Y. Bartal and E. Grove, “The harmonic k -server algorithm is competitive,” *Journal of the ACM*, vol. 47, no. 1, pp. 1–15, 2000.