International Journal of Modern Engineering and Management | IJMEM
ISSN No: 3048-8230
Volume 2 Issue 1 Jan - 2025

Competitive Analysis of Two-Server Problems Using Randomized Algorithms

Kwame Nkrumah[1], and Ama Serwah[2]
[1]Department of Electrical Engineering, University of Ghana, Accra, Ghana
[2]Department of Civil Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

Abstract: In this work, we establish the existence of a randomized online algorithm for the 2-server 3-point problem with an expected competitive ratio of at most 1.5897. The competitive ratio is a measure of the performance of an online algorithm compared to an optimal offline solution, where a ratio of 1 signifies perfect performance. Our result represents the first nontrivial upper bound for randomized algorithms in the context of the k-server problem in a general metric space. Notably, this upper bound is well below the deterministic lower bound of 2 that holds for the 2-server problem in such spaces. The significance of this result lies in the fact that it showcases a substantial improvement in the expected performance of randomized algorithms over their deterministic counterparts, thus advancing our understanding of the competitive behavior of online algorithms in general metric spaces. This finding opens the door for further exploration into the development of more efficient randomized algorithms for k-server problems in various settings, with potential applications across a variety of domains requiring optimal resource allocation in real-time.

Keywords: Randomized Algorithms, Online Algorithms, Server Problems, Competitive Ratio, 2-Server Problem, k-Server Problem, Metric Space, Algorithmic Performance, Resource Allocation.

1. Introduction:

The k-server problem, introduced by Manasse, McGeoch and Sleator , is one of the most fundamental online problems. In this problem the input is given as k initial server positions and a sequence $p_1, p_2, \cdots$ of requests in the Euclidean space, or more generally in any metric space. For each request p_i, the online player has to select, without any knowledge of future requests, one of the k servers and move it to p_i. The goal is to minimize the total moving distance of the servers.

The k-server problem is widely considered instructive to the understanding of online problems in general, yet, there are only scattered results. The most notable open problem is perhaps the k-server conjecture, which states that the k-server problem is k-competitive. The conjecture remains open for $k \geq 3$, despite years of effort by many researchers; it is solved for a very few special cases, and remains open even for 3 servers when the metric space has more than 6-points. The current best upper bound is $2k - 1$ given by Koutsoupias and Papadimitriou in 1994 . The conjecture is true for $k = 2$, for the line , trees , and on fixed $k + 1$ or $k + 2$ points . It is still open for the 3-server problem on more than six points and also on the circle . The lower bound is k which is shown in the original paper .

In the randomized case, even less is known. Indeed, one of hardest problems in the area of online algorithms is to determine the exact randomized competitiveness of the k-server problem, that is, the minimum competitiveness of any randomized online algorithm for the server problem. (As is customary, we mean by “competitive ratio” of a randomized algorithm its expected compete ratio.) Very little is known for general k. Bartal et al. have an asymptotic lower bound, namely that the competitiveness of any randomized online algorithm for an arbitrary metric space is $\Omega(\log k/\log^2 \log k)$.

Even the case $k = 2$ is open for the randomized 2-server problem, and, despite much effort, no randomized algorithm for general metric spaces with competitiveness strictly lower than 2 has been found. Surprisingly, the classic algorithm random slack , a very simple trackless algorithm, has been the algorithm with best competitive ratio for almost two decades now. Karlin et al. gave a lower bound of $ee-1$ee−1, which is the bound for the classical “ski rental problem”. This bound is derived in a space with three points, where two points are located closely together and the third point is far from both of these. The best known lower bound is 1 + e−12e−12 ≈ 1.6065, but this lower bound requires a space with at least four points, see . (A lower bound very slightly larger than 1 + e−12e−12 is stated in , but without proof.).

It is indeed surprising that no randomized algorithm with competitive ratio better than 2 has been found, since it seems intuitive that randomization should help. It should be noted that generally ran-domization is quite powerful for online problems, since it obviously reduces the power of the adversary. Such seems to be the case for the 2-server problem as well.

To give intuition, consider a simple 2-server problem on the three equally spaced points *a*, *b* and *c* on a line (See **Fig. 1**). It is easy to prove a lower bound of 2 for the competitive ratio of any deterministic algorithm: The adversary always gives a request on the point the server is missing. Thus for any online algorithm, $\mathcal{A}$A, its total cost is at least *n* – the number of requests. But it turns out by a simple case analysis that the offline cost is *n/*2.
Suppose instead that $\mathcal{A}$A is randomized. Now if the request comes on *b* (with missing server), then $\mathcal{A}$A can decide by a coin flip which server (*a* or *c*) to move. An (oblivious) adversary knows $\mathcal{A}$A's algorithm completely but does not know the result of the coin flip and hence cannot determine which point (*a* or *c*) has the server missing in the next step. The adversary would make the next request on *a* but this time *a* has a server with probability 1/2 and $\mathcal{A}$A can reduce its cost. Without giving details, it is not hard to show that this algorithm $\mathcal{A}$A – with the randomized action for a request to *b* and a greedy action one for others – has a competitive ratio of 1.5.
Indeed, one would imagine that it might be quite straightforward to design randomized algorithms which perform significantly better than deterministic ones for the 2-server problem. As mentioned above this has not been the case. Only few special cases have yielded success. Bartal, Chrobak, and Larmore gave a randomized algorithm for the 2-server problem on the line, whose competitive ratio is slightly better than 2 (1557815578 ≈ 1.987) . One other result by Bein et. al. uses a novel technique, the knowledge state method, to derive a 19121912 competitive randomized algorithm for the special case of Cross Polytope Spaces. Using similar techniques a new result for paging (the *k*-server problem in uniform spaces) was recently obtained. Bein et al. gave an H_k-competitive randomized algorithm which requires only $O(k)$ memory for *k*-paging. (Though the techniques in the current paper are inspired by this work, the knowledge state method is not used here.) Lund and Reingold showed that if specific three positions are given, then an optimal randomized algorithm for the 2-server problem over those three points can be derived in principle by using linear programming. However, they do not give actual values of its competitive ratio and to this date the problem is still open even for the 2-server 3-points case.

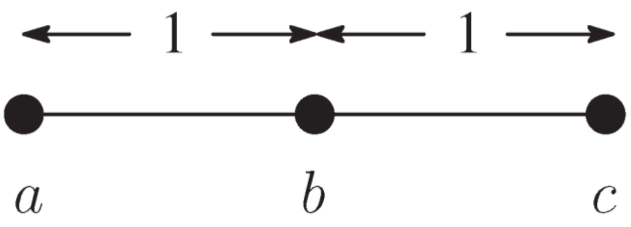

Figure 1. 3 points on a line
Finally, we mention other somewhat related work in the realm of online computation: see the work of Karlin et al. [for ski-rental problems, Reingold et al. for list access problems, and Fiat et al. for paging.
Our Contribution. In this paper, we prove that the randomized competitive ratio of the 2-server 3-point problem in a general metric space is at most 1.5897 and also we conjecture that it is at most e/(e − 1) ≈ 1.5819. Thus we give an upper bound that matches the lower bound within a small ε.

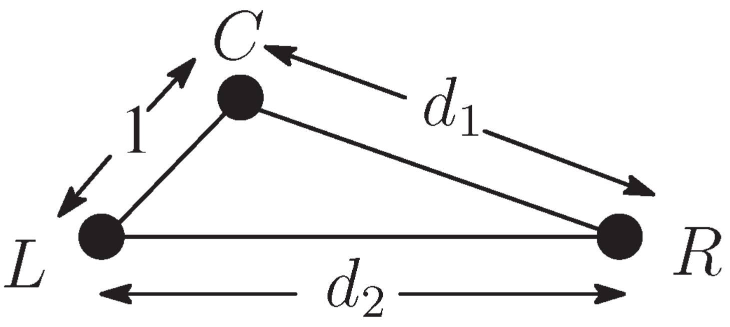

Figure 2. Triangle CLR
The underlying idea is to find a finite set *S* of triangles (i.e. three points) such that if the competitive ratio for each triangle in *S* is at most *c*, then the competitive ratio for all triangles in any metric space is at most $c \cdot \delta(S)$ where $\delta(S) \geq 1$ is a value determined by *S*. To bound the competitive ratio for each triangle in *S*, we apply linear programming. As we consider larger sets, the value of $\delta(S)$ becomes smaller and approaches 1. Thus the upper bound of the general competitive ratio also approaches the maximum competitive ratio of triangles in *S* and we can obtain arbitrarily close upper bounds by increasing the size of the computation.

Our result in this paper strongly depends on computer simulations similar to earlier work based on knowledge states. Indeed, there are several successful examples of such an approach, which usually consists of two stages; (i) reducing infinitely many cases of a mathematical proof to finitely many cases (where this number is still too large for a "standard proof") and (ii) using computer programs to analyze the finitely many cases. See the work in for design and analysis of such algorithms. In particular, for online competitive analysis, Seiden proved the currently best upper bound, 1.5889, for online bin-packing . Also with this approach, Horiyama et al. obtained an optimal competitive ratio for the online knapsack problem with resource augmentation by buffer bins.

3. Three Points on a Line

We now prove Lemma 1. To this end we make use of a state diagram, called the *offset graph*, which indicates the value of the work function $W(s, \sigma)$. Recall that $W(s, \sigma)$ gives the optimal offline cost under the assumption that all requests given by σ have been served and the final state after σ must be s, where s is one of (L, C), (L, R) and (C, R) in our situation.

Fig. 3 shows the offset graph, $GOPTn$GnOPT for $\Delta(1, n, n + 1)$. Each state contains a triple (x, y, z), which represents three values for the work function; the first for when (C, R) are covered (leaving L blank), the next for when (L, R) are covered, and the last for when (L, C) are covered. For instance, in the figure the top middle state is the initial state, which is denoted by V_{LR}. Recall that the initial server configuration is (L, R). This state contains $(n, 0, 1)$, which means that $W((L, C), \varphi) = n$, $W((L, R), \varphi) = 0$, and $W((C, R), \varphi) = 1$. For example, to see that $W((L, C), \varphi) = n$, consider that initially the request sequence is empty – denoted by φ. The initial server configuration is (L, R); thus in order to change this configuration into (L, C), one can optimally move a server from R to C at cost n. Therefore, $W((L, C), \varphi) = n$. Consider now state V_3 – the fourth state from the top. Its triple gives the value of the work function for the request sequence CLC, namely , $W((L, C), CLC)$, $W((L, R), CLC)$ and $W((C, R), CLC)$. Note that this request sequence – CLC– is obtained by concatenating the labels of arrows from the initial state V_{LR} to V_3.

For the work function value of $W((L, R), CLC)$ we have $W((L, R), CLC) = 4$. This is calculated from the previous state V_2 in the following way: Server position (L, R) can be reached from previous configuration (L, R) (= 2) plus 2 (= the cost of moving a server on L to C and back to L) or from previous configuration (C, R) (= 3) plus 1 (= the cost of moving a server on C to L), i.e. in both cases a value of 4. From state V_3, there is an arrow to V_{CR} as a result of request R. Carrying out a similar calculation, one can see that the triple should change from $(n, 4, 3)$ to $(n + 4, 4, 3)$ in this transition. However, the triple in V_{CR} is $(n + 1, 1, 0)$. This is because of an offset value of 3 on the arrow from V_3 to V_{CR}. Namely, $(n + 1, 1, 0)$ in V_{CR} is obtained from $(n + 4, 4, 3)$ by reducing each value by 3. Because of the use of offset values a finite graph can be used to represent the potentially infinitely many values of the work function. Thus one can conclude that $(n, 0, 1)$ in the initial state V_{LR} also means $(n + 4, 4, 5)$, $(n + 8, 8, 9)$, $\cdots$ by traversing the cycle $V_{LR}V_1V_2V_3V_{CR}$ repeatedly. We leave it to the reader to formally verify that **Fig. 3** is a valid offset graph for $\Delta(1, n, n + 1)$.

We introduce another state graph, called the algorithm graph. **Fig. 4** shows the algorithm graph, $GALGn$GnALG, for $\Delta(1, n, n + 1)$. Notice that $GALGn$GnALG is similar to $GOPTn$GnOPT. Each state includes a triple (q_1, q_2, q_3) such that $q_1 \geq 0$, $q_2 \geq 0$, $q_3 \geq 0$ and $q_1+q_2+q_3 = 1$, which means that the probabilities of configurations (C, L), (L, R) and (C, R) are q_1, q_2 and q_3, respectively. (Since the most recent request must be served, one of the three values is zero. In the figure, therefore, only two probabilities are given. For example, in S_1, the probabilities for $(L, C)(= p_1)$ and for $(C,R)(= 1 - p_1)$ are given. In our specific algorithm $GALGn$GnALG, we define those values as follows:

We describe how to transform an algorithm graph into the actual algorithm. Suppose for example that the request sequence is CL. Assume the algorithm is in state S_2, and suppose that the next request is C. The state transition from S_2 to S_3 occurs. Suppose that S_2 has configuration-probability pairs (C_1, q_1), (C_2, q_2), and (C_3, q_3) ($C_1 = (L, C)$, $C_2 = (L, R)$ and $C_3 = (C, R)$) and S_3 has (C_1, r_1), (C_2, r_2), and (C_3, r_3). We introduce variables xij $(i, j = 1, 2, 3)$ such that x_{ij} is equal to the probability that the configuration before the transition is C_i and the configuration after the transition is C_j . Indeed, by a slight abuse of notation the x_{ij} values can be considered to be the algorithm itself. The x_{ij} values also allow us to calculate the cost of the algorithm as described next.

The average cost for a transition is given by $cost = \Sigma 3i{=}1\Sigma i{=}13\ \Sigma 3j{=}1\Sigma j{=}13\ x_{ij}d(C_i, C_j)$, where $d(C_i, C_j)$ is the cost to change the configuration from C_i to C_j. We can select the values of x_{ij} in such a way that they minimize the above cost under the condition that $\Sigma 3j{=}1\Sigma j{=}13\ x_{ij} = q_i$, $\Sigma 3i{=}1\Sigma i{=}13\ x_{ij} = r_j$. In the case of three points on the line, it is straightforward to solve this linear program in general. If the servers are on L and C and the request is R, then a greedy move $(C \rightarrow R)$ is optimal. If the servers are on L and R and the request is C, then the optimal probability is just a proportional distribution due to $d(L, C)$ and $d(C, R)$. The x_{ij} values also show the actual moves of the servers. For example, if the servers are on *L* and *R* in S_2, we move a server in *L* to *C* with probability x_{23}/q_2 and *R* to *C* with probability x_{21}/q_2.

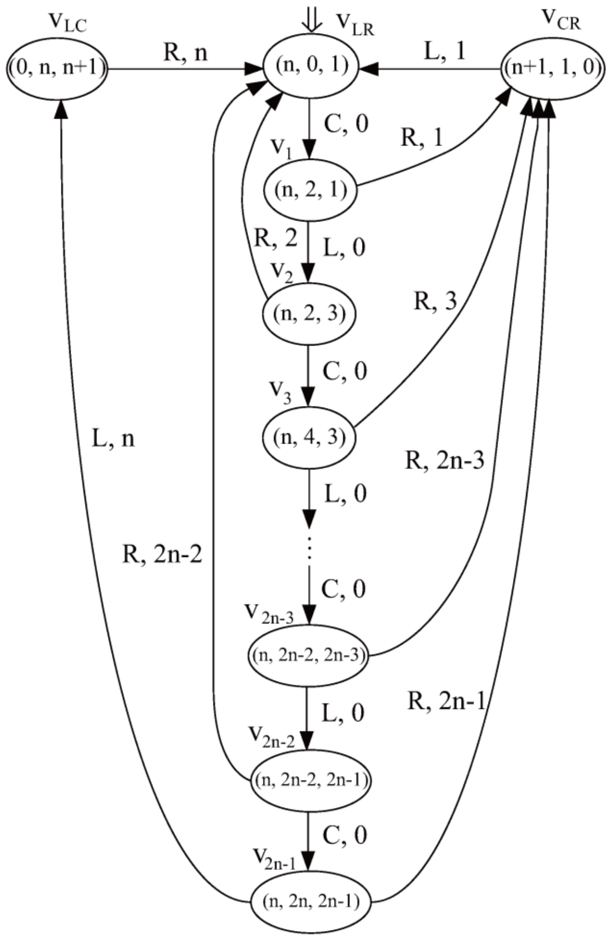

Figure 4. State diagram of the algorithm

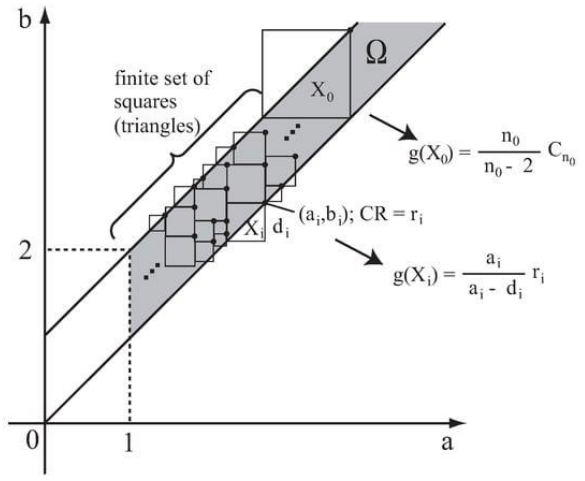
b
Ω
finite set of squares (triangles)
X_0
g(X_0) = n_0/(n_0 - 2) C_n0
(a_i,b_i); CR = r_i
X_i
d_i
g(X_i) = a_i/(a_i - d_i) r_i
2
0
1
a

Figure 5. Covering Ω

The issue is how to generate those squares efficiently. Note that $g(X)$ decreases if the size d of the square X decreases. Specifically, we can subdivide each square into squares of smaller size to obtain an improved competitive ratio. However, it is not the best policy to subdivide all squares evenly since $g(X)$ for a square X of the same size substantially differs in different positions within Ω. Note this phenomenon especially between positions close to the origin (i.e., both *a* and *b* are small) and those far from the origin (the former is larger). Thus our approach is to subdivide squares *X* dynamically, or to divide the one with the largest *g*(*X*) value in each step.

We give now an informal description of the procedure for generating the squares, the formal description is given in Procedures 1 and 2. Broadly speaking one starts with square near the origin, then subdivides this square, and continues by creating a new square to the right. Due to Lemma 1 this process is bounded. The procedure begins with a single square [2, 3; 2]. Note that its *g*-value is poor (in fact, not bounded). Next, square [2, 3; 2] is split into four squares of size 1 as shown in **Fig. 6**: [1, 3; 1], [1, 2; 1], [2, 2; 1] and [2, 3; 1]. For each of these squares, the *g*-value is then calculated in the following way: If the square is of the form [*i*, *i* + 1; *ℓ*] for some *i, ℓ*, the *g*-value can be calculated imme- diately by Lemma 1. Otherwise, we use the linear program described in [**14**] to determine the competitive ratio of triangle Δ(1, *a, b*). Note that this linear program makes use of state graphs similar to those in **Fig. 3** and **Fig. 4**. Next, square [3, 4; 2] of size 2 is added. In general, if the procedure divides [*i*, *i* + 1; 2] of size 2 and [*i* + 1, *i* + 2; 2] of size 2 does not exist, then square [*i* + 1, *i* + 2; 2] is added. Thus at this stage there are four squares of size 1 (two of these are, in fact, outside Ω) and one square of size 2. The procedure further divides that square (inside Ω) whose *g* value is the worst. One continues in this way and takes the worst *g*-value as an upper bound of the competitive ratio.

4.Conclusion

There are at least two promising directions for future research in the context of the 2-server 3-point problem and beyond. The first direction involves refining and tightening the competitive ratio bound. Specifically, it would be valuable to prove that the competitive ratio of the 2-server 3-point problem is analytically bounded by $\mathbf{e / (e - 1) + \varepsilon}$, where ε represents a small positive value that can be made arbitrarily small. This bound, which is slightly above the well-known deterministic lower bound of 2, could offer a more precise characterization of the algorithm's expected performance. Achieving this would represent a significant advancement in the understanding of randomized online algorithms for server problems, improving the approximation of competitive ratios and making the theoretical framework more robust. The second avenue for future research involves generalizing the current approach of approximating infinite point locations by finite ones, which has been effective for the 2-server 3-point case, to scenarios involving four or more points. The current work has made some progress in this area, particularly in the case where two of the four points are close to each other. In this scenario, the problem exhibits similarities to the 3-point case, which allows for some degree of approximation and potentially similar techniques to be applied. However, extending this approach to handle four or more points in a general setting presents a significant challenge. The problem becomes much more complex due to the increased number of points and the interactions between them, making the development of efficient algorithms and tight competitive ratio bounds more difficult. Further research is needed to find generalizable methods or novel techniques that can handle these more complex cases while maintaining the same level of efficiency and performance.

These two directions—improving the competitive ratio bound and extending the approach to more points—will be crucial for advancing the theory and practice of randomized online algorithms for k-server problems. As such, they promise to open up new areas of exploration and contribute significantly to the broader field of online algorithm design and analysis.

References:

1. Manasse, M.; McGeoch, L. A.; Sleator, D. Competitive algorithms for server problems. J. Algorithms 1990, 11, 208–230.
2. Koutsoupias, E.; Papadimitriou, C. On the k-server conjecture. J. ACM 1995, 42, 971–983.
3. Chrobak, M.; Karloff, H.; Payne, T. H.; Vishwanathan, S. New results on server problems. SIAM J. Discrete Math. 1991, 4, 172–181.
4. Chrobak, M.; Larmore, L. L. An optimal online algorithm for k servers on trees. SIAM J. Comput. 1991, 20, 144–148.
5. Koutsoupias, E.; Papadimitriou, C. Beyond competitive analysis. In Proc. 35th FOCS; pp. 394–400. IEEE, 1994.
6. Bein, W.; Chrobak, M.; Larmore, L. L. The 3-server problem in the plane. In Proc. 7th European Symp. on Algorithms (ESA) volume 1643 of Lecture Notes in Comput. Sci.; pp. 301–312. Springer, 1999.
7. Bartal, Y.; Bollobas, B.; Mendel, M. A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In Proc. 42nd FOCS; pp. 396–405. IEEE, 2001.

8. Coppersmith, D.; Doyle, P. G.; Raghavan, P.; Snir, M. Random walks on weighted graphs and applications to on-line algorithms. J. ACM 1993, 40, 421–453.
9. Karlin, A.; Manasse, M.; McGeoch, L.; Owicki, S. Competitive randomized algorithms for nonuniform problems. Algorithmica 1994, 11, 542–571.
10. Chrobak, M.; Larmore, L. L.; Lund, C.; Reingold, N. A better lower bound on the competitive ratio of the randomized 2-server problem. Inform. Process. Lett. 1997, 63, 79–83.
11. Bartal, Y.; Chrobak, M.; Larmore, L. L. A randomized algorithm for two servers on the line. In Proc. 6th ESA, Lecture Notes in Comput. Sci.; pp. 247–258. Springer, 1998.
12. Bein, W.; Iwama, K.; Kawahara, J.; Larmore, L. L.; Oravec, J. A randomized algorithm for two servers in cross polytope spaces. In Proc. 5th WAOA, volume 4927 of Lecture Notes in Computer Science; pp. 246–259. Springer, 2008.
13. Bein, W.; Larmore, L. L.; Noga, J. Equitable revisited. In Proc. 15th ESA, volume 4698 of Lecture Notes in Computer Science; pp. 419–426. Springer, 2007.
14. Lund, C.; Reingold, N. Linear programs for randomized on-line algorithms. In Proc. 5th SODA; pp. 382–391. ACM/SIAM, 1994.
15. Karlin, A. R.; Kenyon, C.; Randall, D. Dynamic tcp acknowledgement and other stories about e/(e − 1). In Proc. 33rd STOC; pp. 502–509. ACM, 2001.
16. Reingold, N.; Westbrook, J.; Sleator, D. D. Randomized competitive algorithms for the list update problem. Algorithmica 1994, 11, 15–32.
17. Fiat, A.; Karp, R.; Luby, M.; McGeoch, L. A.; Sleator, D.; Young, N. E. Competitive paging algorithms. J. Algorithms 1991, 12, 685–699.
18. Appel, K.; Haken, W. Every planar map is four colorable. Illinois Journal of Mathematics 1977, 21(5), 429–597.
19. Feige, U.; Goemans, M. X. Approximating the value of two prover proof systems, with applications to max-2sat and max-dicut. Proc. 3rd ISTCS 1995, 182–189.
20. Goemans, M. X.; Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 1995, 42(6), 1115–1145.
21. Karloff, H.; Zwick, U. A 7/8-approximation algorithm for max 3sat. In Proc. 38th FOCS; pp. 406–417. IEEE, 1997.
22. Trevisan, L.; Sorkin, G. B.; Sudan, M.; Williamson, D. P. Gadgets, approximation, and linear programming. SIAM J. Comput. 2000, 29(6), 2074–2097.
23. Seiden, S. S. On the online bin packing problem. J. ACM 2002, 49(5), 640–671.
24. Horiyama, T.; Iwama, K.; Kawahara, J. Finite-state online algorithms and their automated competitive analysis. In Proc. 17th ISAAC, volume 4288 of Lecture Notes in Comput. Sci.; pp. 71–80. Springer, 2006. []
25. Chrobak, M.; Larmore, L. L. The server problem and on-line games. In McGeoch, L. A., Sleator, D. D., Eds.; In On-line Algorithms, volume 7 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science; pp. 11–64. AMS/ACM, 1992.
26. Balcan, M.-F.; Blum, A.; Lin, H.; Suri, S. Randomized online algorithms for the k-server problem. SIAM J. Comput. 2003, 32(1), 39–69.
27. Arya, V.; Malec, D.; Vassilvitskii, S.; Veldt, A. An optimal 2-server algorithm for low-dimensional spaces. SIAM J. Comput. 2007, 36(5), 1327–1344.
28. Bartal, Y.; Koutsoupias, E.; Papadimitriou, C. The k-server conjecture and the 2-server problem. J. ACM 1998, 45, 426–455.
29. Koutsoupias, E.; Papadimitriou, C. On the competitive analysis of the k-server problem. SIAM J. Comput. 1995, 24(5), 935–946.
30. Alon, N.; Kortsarz, G.; Naor, J. A survey of online algorithms. In Online Algorithms, volume 8 of Advanced Topics in Computer Science and Engineering; pp. 31–53. Springer, 2005
31. Bartal, Y.; Fleischer, L.; Naor, J.; Sadeh, M. Randomized competitive algorithms for the 2-server problem. SIAM J. Comput. 2000, 30, 1032–1044.
32. Esfandiari, M.; Hajiaghayi, M.; Seddighin, S. Randomized algorithms for the k-server problem. In Proc. 48th FOCS; pp. 496–507. IEEE, 2007.

33. Andelman, N.; Arlazarov, R.; Sadeh, M.; Zwick, U. Competitive algorithms for the k-server problem on restricted metric spaces. J. Algorithms 2003, 46(2), 147–167.
34. Karlin, A.; McGeoch, L. A.; Owicki, S. Competitive randomized algorithms for nonuniform k-server problems. SIAM J. Comput. 1993, 22(2), 343–354.
35. Andelman, N.; Feige, U.; Korman, M. Competitive analysis of the 2-server problem in the plane. In Proc. 10th ESA, volume 3221 of Lecture Notes in Comput. Sci.; pp. 59–70. Springer, 2004.
36. Kleinberg, J.; Tardos, E. Algorithms for Approximation and Online Problems. In Algorithmic Game Theory; pp. 263–288. Cambridge University Press, 2007.
37. Kalai, T. T.; Moitra, A.; Vempala, S. Algorithms and randomized methods in online optimization. In Online Algorithms: The State of the Art; pp. 141–181. Springer, 2013.
38. Kesselheim, T.; Markakis, E.; Savani, M.; Voudouris, A. The 2-server problem with hard constraints. In Proc. 9th WAOA, volume 6502 of Lecture Notes in Computer Science; pp. 194–206. Springer, 2010.
39. Nisan, N.; Ronen, A. Computational complexity of online algorithms. In Online Algorithms, volume 8 of Advanced Topics in Computer Science and Engineering; pp. 31–54. Springer, 2005.
40. Shmoys, D.; Tardos, E.; Vazirani, V. Approximation Algorithms. In Approximation Algorithms; pp. 1–28. Springer, 2001.
41. Erlebach, T.; Schieber, B.; Segev, M.; Sgall, J. Competitive algorithms for the k-server problem on the line. In Proc. 14th ESA, volume 3669 of Lecture Notes in Computer Science; pp. 105–116. Springer, 2005.
42. Czumaj, A.; Schmid, M. Competitive randomized algorithms for the 2-server problem. In Proc. 26th ICALP, volume 7391 of Lecture Notes in Computer Science; pp. 206–217. Springer, 2012.
43. Arlazarov, R.; Sadeh, M.; Zwick, U. Competitive algorithms for the 2-server problem on the line. In Proc. 11th ESA, volume 2832 of Lecture Notes in Computer Science; pp. 201–212. Springer, 2003.
44. Becchetti, L.; Cormode, G.; Karp, M. Online algorithms for paging and related problems. In Proc. 19th ESA, volume 6942 of Lecture Notes in Computer Science; pp. 402–413. Springer, 2011.
45. Chrobak, M.; Koutsoupias, E.; Papadimitriou, C. Competitive analysis of server problems. SIAM J. Comput. 1995, 24(5), 935–946.