Journal of Computing and Information Technology - CIT 18, 2010, 4, 361–367
doi:10.2498/cit.1001922

Measuring True Performance of the Work Function Algorithm for Solving the On-line *k*-Server Problem

Tomislav Rudec[1], Alfonzo Baumgartner[1] and Robert Manger[2]

[1]Faculty of Electrical Engineering, University of Osijek, Croatia
[2]Department of Mathematics, University of Zagreb, Croatia

This paper deals with the work function algorithm (WFA) for solving the on-line *k*-server problem. The paper is concerned with assessing actual performance of the WFA in terms of its serving costs. First, an efficient implementation of the WFA is briefly described. Next, some experiments are presented, where the performance of the implemented WFA has been measured on very large problem instances. Thereby, the problem instances have been selected from some frequently studied classes where sharp theoretical estimates of performance are available. Finally, the measured performance of the WFA is compared with the corresponding theoretical estimates and with some other algorithms.

Keywords: on-line problems, on-line algorithms, *k*-server problem, work function algorithm (WFA), experiments

1. Introduction

In the *k*-server problem [11] one has to decide how *k* mobile servers should serve a sequence of requests appearing at various locations of a metric space with altogether *m* locations. It is usually required that the solution is produced in on-line fashion [6], so that each request is served before the next request arrives. Serving is accomplished by moving a server to the appropriate location. In addition to processing requests on time, a good on-line algorithm for solving the *k*-server problem also tries to minimize the total cost of serving, where the cost is estimated as the total distance crossed by all servers. A desirable property of such quasi-optimal serving is called *competitiveness* [14]. Vaguely speaking, an algorithm is competitive if its cost is only a bounded number of times worse than optimal.

Among various on-line algorithms for solving the *k*-server problem, the best characteristics regarding competitiveness exhibit the *work function algorithm*(WFA) [4]. Still, in spite of its interesting theoretical properties, the WFA has seldomly been used in practice due to its prohibitive computational complexity. Thus, with lack of practical evidence, it is not quite clear whether the competitive, but complex WFA can really provide better service than much simpler, but non-competitive heuristics [6,11].

In a recent paper [13] we developed a new implementation of the WFA, which is fast enough to run on very large request sequences and with many servers. Thanks to our implementation, it has become possible to measure experimentally actual serving costs of the WFA and compare it with theoretical estimates and/or other algorithms.

In [13] we presented only the most important experimental results based on our implementation of the WFA. Now we concentrate on additional results, which refer to some special cases where the WFA achieves the best possible competitiveness bound. Thus, the aim of this paper is to assess actual performance of the WFA in situations where the most accurate theoretical estimates of performance are available. In such situations it is possible to compare the measured performance with the corresponding estimates and check if they are in accordance or not. Consequently, the paper is also indirectly concerned with practical relevance of competitiveness theory in general.

The text is organized as follows. After the introduction, all necessary preliminaries are listed in Section 2. Section 3 briefly describes our implementation of the WFA. Section 4 presents the chosen experimental results. The final Section 5 gives the conclusion.

2. Preliminaries

An instance of the k-server problem is given by the initial configuration of k servers $S^{(0)} = (s_1^{(0)}, s_2^{(0)}, \ldots, s_k^{(0)})$, where $s_j^{(0)}$ specifies the initial location of the j-th server, and with the sequence of n requests $\sigma = (r_1, r_2, \ldots, r_n)$, where r_i determines the location of the i-th request. In its i-th step, an on-line algorithm for solving the k-server problem serves the request r_i by moving a server to the location of r_i. Thereby the current server configuration $S^{(i-1)}=(s_1^{(i-1)}, s_2^{(i-1)}, \ldots, s_k^{(i-1)})$ transforms into a new configuration $S^{(i)} = (s_1^{(i)}, s_2^{(i)}, \ldots, s_k^{(i)})$. The decision which server to move may be based only on the requests already seen r_1, r_2, $\ldots$, r_i. Whenever the algorithm moves a server from a location a to a location b, it incurs a cost equal to the distance $D(a, b)$ among a and b.

As a concrete instance of the k-server problem, let us consider the space of $m = 4$ locations shown in Figure 1 with distances given. Suppose that $k = 3$ servers are initially located at locations A, B and D. If the first request appears in C, then our on-line algorithm has to decide which of the servers should be moved to that location. One possibility is to move the nearest server from A. But maybe it is better to move the farthest server from D — such move would be optimal if all forthcoming requests appeared in A, B and C, and none in D.

The simplest on-line method for solving the k-server problem is the *random algorithm* (RAND)

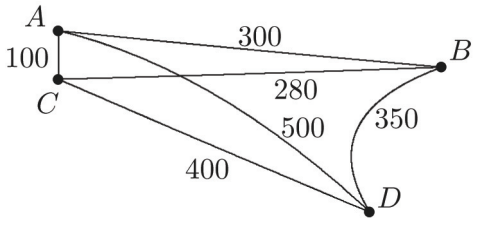

Figure 1. A k-server problem instance.

[6], where each request is served by a randomly chosen sever. The most popular heuristic is the *greedy algorithm* (GREEDY) [6], which serves the current request by sending the nearest server to the requested location. Another heuristic is the *balanced algorithm* (BALANCE) [11], which attempts to keep distances moved by various servers roughly equal, thus employing the server whose cumulative distance traveled so far, plus the distance to the requested location, is minimal.

This paper is concerned with the *work function algorithm* (WFA) [4]. In its i-th step the WFA serves the request r_i by switching from the current server configuration $S^{(i-1)}$ to a new configuration $S^{(i)}$ chosen so that

$$F(S^{(i)}) = C_{\mathrm{OPT}}(S^{(0)}, r_1, r_2, \ldots, r_i, S^{(i)}) + D(S^{(i-1)}, S^{(i)})$$

becomes minimal. Thus the objective function $F(S^{(i)})$ is defined here as a sum of two parts. The first part is the minimum total cost of starting from $S^{(0)}$, serving in turn $r_1, r_2, \ldots, r_i$, and ending up in $S^{(i)}$. The second part is the distance traveled by a server to switch from $S^{(i-1)}$ to $S^{(i)}$.

Now we give a formal definition of competitiveness [14]. Let ALG be an on-line algorithm. Let OPT be the optimal off-line algorithm that knows the whole input in advance and serves the whole request sequence at minimum total cost. Denote with $C_{\mathrm{ALG}}(S^{(0)}, \sigma)$ the total cost incurred by ALG on the problem instance given by the initial server configuration $S^{(0)}$ and the request sequence σ. Denote with $C_{\mathrm{OPT}}(S^{(0)}, \sigma)$ the minimum total cost on the same input data. For a given constant α, we say that ALG is α-*competitive* if there exists another constant β such that on every $S^{(0)}$ and every σ it holds:

$$C_{\mathrm{ALG}}(S^{(0)}, \sigma) \leq \alpha \cdot C_{\mathrm{OPT}}(S^{(0)}, \sigma) + \beta.$$

Finally, we review some important theoretical results dealing with competitiveness. It can easily be proved [11] that any α-competitive algorithm for the k-server problem must have $\alpha \geq k$. Also, it is easy to check [6] that RAND, GREEDY and BALANCE are not competitive.

Finally, it has been proved in [8,10] that the WFA is $(2k-1)$-competitive. It is believed that the WFA is in fact k-competitive, thus achieving the best possible bound. However, this hypothesis has been proved only for some special cases, e.g. if the locations form a line [1,3], if all but two locations are covered by servers [9], or if there are exactly two servers [11].

3. Implementation

Our implementation of the WFA is based on the network flow techniques [2], and it combines the following three features:

- a compact network model, which reduces each step of the WFA to only one minimal-cost maximal flow problem instance;
- a customized algorithm for computing optimal network flows, which takes into account special properties of the involved networks;
- parallelization of most demanding parts of the algorithm, so that the workload is distributed among several cores of a multi-core processor.

Our network model corresponding to the i-th step of the WFA is shown in Figure 2. Thus the network consists of a source $\bar{s}$, a sink $\bar{t}$, and four additional layers of nodes. The first layer represents the initial server configuration $S^{(0)}$, i.e. each s_j $(j = 1, 2, \ldots, k)$ corresponds to the starting location of one server. The second and third layers together represent the request sequence; thereby both r_p and r'_p $(p = 1, 2, \ldots, i)$

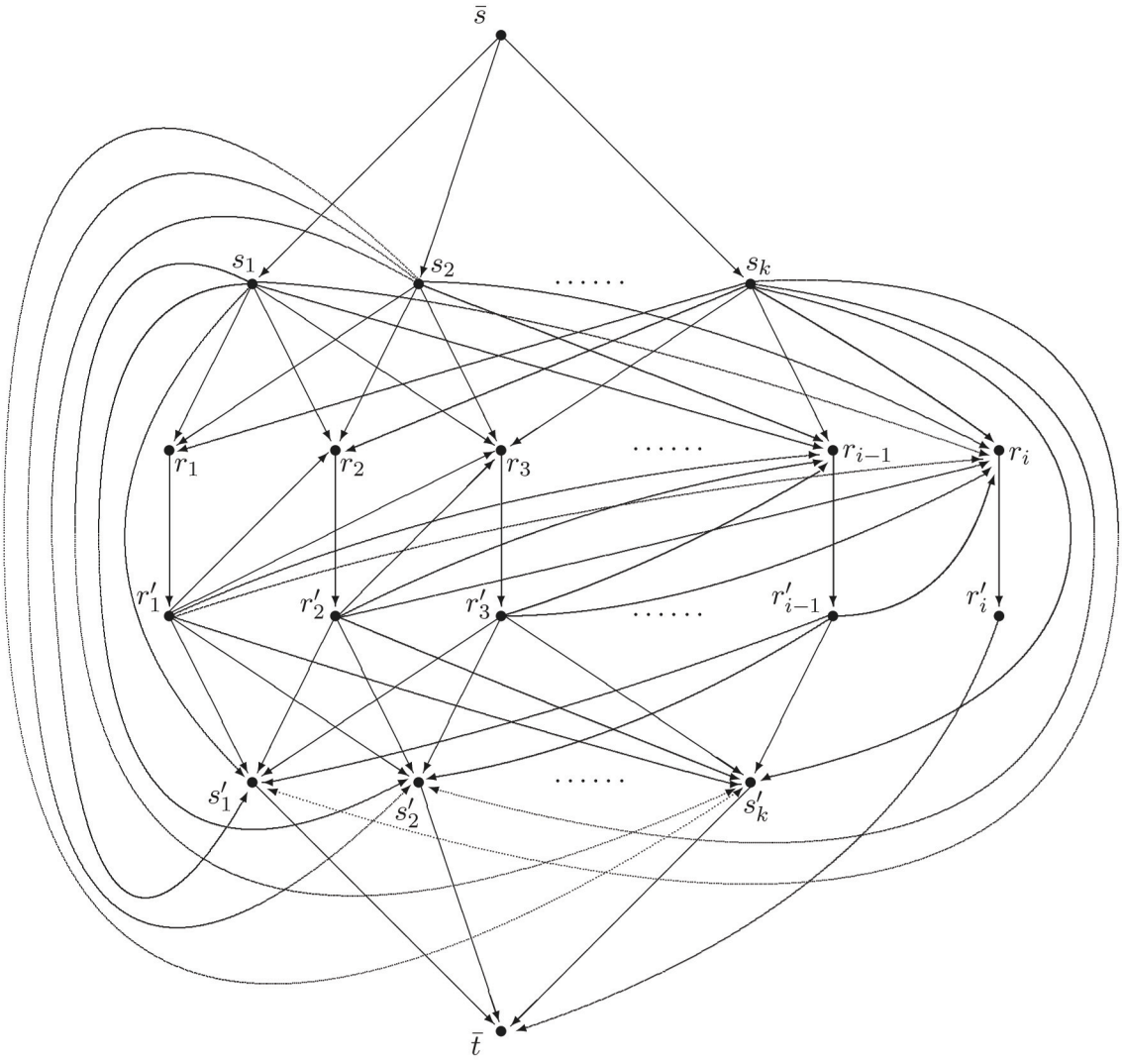

Figure 2. The network corresponding to the i-th step of the WFA.

correspond to the location of the p-th request. The fourth layer specifies the current server configuration $S^{(i-1)}$, i.e. each s'_j $(j = 1, 2, \ldots, k)$ corresponds to a location covered by a server immediately before the i-th step of the WFA.

All arcs in the network shown in Figure 2 have unit capacities. The arcs leaving $\bar{s}$ or r'_i have zero costs. The cost of an arc $r_p \rightarrow r'_p$ is $-L$, where L is a suitably chosen very large positive number. The cost of an arc $s'_l \rightarrow \bar{t}$ is

$$\frac{1}{k-1} \sum_{q=1}^{k} D(s'_q, r_i) - D(s'_l, r_i).$$

The remaining arcs have costs equal to distances $D(\ ,\ )$ among the corresponding locations.

The i-th step of the WFA is implemented by finding the minimal-cost maximal flow in the network from Figure 2, by determining the unique node s'_x in the fourth layer that is not saturated by the computed flow, and by sending the server from the location specified by s'_x to the location of the i-th request r_i. The correctness proof is given in [13].

Computing the minimal-cost maximal flow in the network from Figure 2 is based on a general *flow augmentation* method [2]. The method reduces to exactly k iterations, and each of those iterations reduces to a single-source shortest path problem instance in a so-called displacement network. With some preprocessing of arc costs [5], any shortest path problem instance can be solved by Dijkstra's procedure [7].

Our customized algorithm has been obtained from the general flow augmentation method by modifying each of its iterations separately. Thus, each iteration of the algorithm tries to find the required shortest path faster than it would be possible with the standard Dijkstra's procedure. All modifications rely on special properties of particular displacement networks. Here are some details.

- In each iteration arcs or nodes are identified, which cannot influence the remaining part of the algorithm. Such arcs or nodes are removed from the network.
- In the first iteration, it is observed that the shortest path sought is almost completely determined in advance, only one arc has yet to be chosen. Consequently, the shortest path is constructed directly, by considering k possibilities.
- In the second iteration, it is observed that the displacement network is acyclic. Thus the shortest path is found by scanning of nodes in topological order, which is much faster than with Dijkstra.
- In the remaining iterations, the shortest path is found by a slightly customized Dijkstra, with preprocessing of arc costs.

More details about each iteration can be found in [12], where an analogous implementation of OPT has been described.

The most demanding parts of the algorithm that are being parallelized are preprocessing of arc costs prior to the application of Dijkstra's procedure, and the Dijkstra's procedure itself. For a moderate number of cores up to 8, the obtained speedup is quite satisfactory.

4. Experiments

In order to perform experiments, we realized our implementation of the WFA as a C++ program. To allow comparison, we also realized two heuristics, GREEDY and BALANCE, as well as the random algorithm RAND and the optimal off-line algorithm OPT. All programs are available for download from our web site `http://art.etfos.hr`. The realized software has been used in various experiments, where the same problem instances have been solved by different algorithms for comparison. The full list of obtained results is constantly updated, and it can also be found on the above mentioned web site.

In this paper we present only the experiments based on problem instances similar to those studied in [1,3,9]. Thus, we consider metric spaces whose locations form a line, or situations where the number of servers k is equal to the number of possible locations m minus 2. Distances are always computed as Euclidean distances. In all problem instances, the request sequence has the length $n = 10000$, and the

initial configuration of servers is chosen so that the servers are evenly spread through the space.

In our experiments, special consideration has been given to distribution of requests among locations, which can be uniform or non-uniform. Non-uniform distribution means that certain locations occur more frequently than the others, thus allowing the WFA to learn from "history".

The results of the experiments are summarized in Tables 1 and 2. The first table refers to metric spaces where the locations are arranged equidistantly along a line. The second table involves metric spaces whose locations are organized as equidistant square grids, and where $k = m - 2$.

Within each table, a column corresponds to problem instances with a given combination of k and m. Actually, there are exactly two such instances, and they differ only in the way how their requests are distributed among locations - uniformly or non-uniformly. A table row corresponds to a particular algorithm: RAND, GREEDY, BALANCE, and the WFA, respectively.

A single table entry contains two values, and it records the performance (total cost) of the corresponding algorithm on the corresponding two problem instances. The upper value refers to uniform distribution of requests, and the lower value in brackets to non-uniform distribution. In both cases, performance is expressed relatively, as the ratio between the cost incurred by the corresponding algorithm and the optimal cost incurred by OPT. In this way, we obtain some kind of empirical measurement of competitiveness.

From the data presented in Tables 1 and 2 the following facts can be observed.

- The measured competitiveness bound of the WFA is between 1 and 3, which is much lower than k suggested by theory.

Number of servers (k)	2			5			100		
Number of locations (m)	3	4	1000	6	10	1000	101	200	1000
$\frac{\text{cost of RAND}}{\text{cost of OPT}}$	1.66 (1.74)	1.77 (1.91)	2.23 (2.48)	3.66 (4.10)	4.11 (4.30)	6.20 (6.39)	61.0 (60.0)	47.0 (28.0)	50.0 (50.0)
$\frac{\text{cost of GREEDY}}{\text{cost of OPT}}$	1.25 (1.24)	1.12 (1.10)	1.08 (1.08)	1.58 (1.64)	1.24 (1.30)	1.15 (1.15)	1.81 (1.80)	2.10 (2.03)	2.77 (2.77)
$\frac{\text{cost of BALANCE}}{\text{cost of OPT}}$	1.25 (1.24)	1.27 (1.28)	1.34 (1.35)	1.58 (1.64)	1.54 (1.60)	1.65 (1.66)	1.81 (1.40)	1.98 (2.04)	2.04 (2.04)
$\frac{\text{cost of the WFA}}{\text{cost of OPT}}$	1.25 (1.06)	1.14 (1.09)	1.10 (1.08)	1.58 (1.49)	1.39 (1.30)	1.17 (1.16)	1.76 (1.20)	1.49 (1.28)	1.68 (1.65)

Table 1. Experimental results – locations form a line – requests distributed (non-)uniformly.

Number of locations (m)	4	5	10	50	100
$\frac{\text{cost of RAND}}{\text{cost of OPT}}$	1.61 (1.87)	1.96 (2.12)	3.53 (3.41)	10.0 (10.2)	15.2 (13.5)
$\frac{\text{cost of GREEDY}}{\text{cost of OPT}}$	1.22 (1.24)	1.46 (1.67)	1.97 (2.14)	2.75 (2.76)	2.65 (3.01)
$\frac{\text{cost of BALANCE}}{\text{cost of OPT}}$	1.38 (1.47)	1.61 (1.71)	2.15 (2.15)	3.55 (3.19)	2.78 (2.59)
$\frac{\text{cost of the WFA}}{\text{cost of OPT}}$	1.30 (1.17)	1.58 (1.65)	2.23 (2.09)	3.02 (2.76)	3.16 (2.48)

Table 2. Experimental results – all but 2 locations covered by servers – requests distributed (non-)uniformly.

- Performance of the WFA is similar to GREEDY, or even slightly better if the distribution of requests among locations is non-uniform.
- The WFA is usually better than BALANCE, and always better than RAND.

5. Conclusion

In this paper we have studied the work function algorithm (WFA) for solving the on-line k-server problem. Although the WFA is very interesting from the theoretical point of view, its true performance in terms of serving costs is hard to assess due to its extremely large computational complexity. With this paper, we are trying to give a modest contribution to such an assessment.

The experimental results presented in the paper clearly show that, at least for the considered classes of problem instances, the actual performance of the WFA is in fact much better than the corresponding theoretical estimates would suggest, although those estimates are already on their best possible limits. Also, the results demonstrate that the WFA performs similarly or only slightly better than a simple heuristic such as the greedy algorithm, although according to theory it should perform much better. Thus, to summarize, the true performance of the WFA is quite different from what one would expect according to theory.

From a more general perspective, the results of this paper clearly confirm that the available theoretical estimates about the WFA and related algorithms do not reflect their typical behavior, but only their pathological worst cases. Consequently, there is an urgent need to devise some other theoretical mechanisms apart from competitiveness, which would allow more accurate ranking of on-line algorithms.

References

[1] Y. BARTAL, E. KOUTSOUPIAS, On the competitive ratio of the work function algorithm for the k-server problem. *Theoretical Computer Science*, 324 (2004), pp. 337–345.

[2] M.S. BAZARAA, J.J. JARVIS, H.D. SHERALI, *Linear Programming and Network Flows*. Third edition, Wiley, New York, 2004.

[3] M. CHROBAK, H. KARLOFF, T.H. PAYNE, S. VISHWANATHAN, New results on server problems. *SIAM Journal on Discrete Mathematics*, 4 (1991), pp. 172–181.

[4] M. CHROBAK, L.L. LARMORE, The server problem and online games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7 (1992), pp. 11–64.

[5] J. EDMONDS, R.M. KARP, Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19 (1972), pp. 248–264.

[6] S. IRANI, A.R. KARLIN, Online computation. In: *D. Hochbaum, editor; Approximation Algorithms for NP-Hard Problems*, PWS, Boston MA, (1997), pp. 521–564.

[7] D. JUNGNICKEL, *Graphs, Networks and Algorithms*. Second edition, Springer, Berlin, 2005.

[8] E. KOUTSOUPIAS, C. PAPADIMITROU, On the k-server conjecture. *Journal of the ACM*, 42 (1995), pp. 971–983.

[9] E. KOUTSOUPIAS, C. PAPADIMITROU, The 2-evader problem. *Information Processing Letters*, 57 (1996), pp. 249–252.

[10] E. KOUTSOUPIAS, Weak adversaries for the k-server problem. In: *P. Beame, editor; The 40th Annual Symposium on Foundations of Computer Science 1999 Oct 17–18*, New York. IEEE, (1999), pp. 444–449.

[11] M. MANASSE, L.A. MCGEOCH, D. SLEATOR, Competitive algorithms for server problems. *Journal of Algorithms*, 11 (1990), pp. 208–230.

[12] T. RUDEC, A. BAUMGARTNER, R. MANGER, A fast implementation of the optimal off-line algorithm for solving the k-server problem. *Mathematical Communications*, 14 (2009), pp. 123–138.

[13] T. RUDEC, A. BAUMGARTNER, R. MANGER, A fast work function algorithm for solving the k-server problem. Submitted for publication, 2009.

[14] D. SLEATOR, R.E. TARJAN, Amortized efficiency of list update and paging rules. *Comm of the ACM*, 28 (1985), pp. 202–208.

Received: June, 2010
Accepted: November, 2010

Contact addresses:
Tomislav Rudec
Faculty of Electrical Engineering
University of Osijek
Kneza Trpimira 2b
31000 Osijek
Croatia
e-mail: Tomislav.Rudec@etfos.hr

Alfonzo Baumgartner
Faculty of Electrical Engineering
University of Osijek
Kneza Trpimira 2b
31000 Osijek
Croatia
e-mail: Alfonzo.Baumgartner@etfos.hr

Robert Manger
Department of Mathematics
University of Zagreb
Bijenička cesta 30
10000 Zagreb
Croatia
e-mail: Robert.Manger@math.hr

TOMISLAV RUDEC received the MSc degree (2001) in mathematics from the University of Zagreb. Currently, he is a lecturer at the Faculty of Electrical Engineering, Josip Juraj Strossmayer University in Osijek. His research interests include combinatorial optimization, analysis of algorithms, on-line computation, and teaching of mathematics. He has published 5 scientific papers in international journals or in conference proceedings, and 6 professional papers.

ALFONZO BAUMGARTNER received the PhD degree (2010) in computer science from the Josip Juraj Strossmayer University in Osijek, Croatia. Currently, he works as an assistant professor at the Faculty of Electrical Engineering, Josip Juraj Strossmayer University in Osijek. His research interests include on-line problems, parallel algorithms applied to combinatorial optimization problems, and other algorithms which use efficient data structures. He has published 7 papers in international scientific journals or in conference proceedings.

ROBERT MANGER received the BSc (1979), MSc (1982), and PhD (1990) degrees in mathematics, all from the University of Zagreb. For more than ten years he worked in industry, where he obtained experience in programming, computing, and designing information systems. Dr Manger is presently a professor at the Department of Mathematics, University of Zagreb. His current research interests include: combinatorial optimization, parallel and distributed algorithms, and soft computing. He has published 20 papers in international scientific journals, over 25 scientific papers in conference proceedings, 10 professional papers, and 4 course materials. Dr Manger is a member of the Croatian Mathematical Society, Croatian Society for Operations Research and IEEE Computer Society.