Journal of Computing and Information Technology - CIT 15, 2007, 4, 325–330
doi:10.2498/cit.1001140

Work Function Algorithm with a Moving Window for Solving the On-line *k*-server Problem

Alfonzo Baumgartner[1], Robert Manger[2] and Željko Hocenski[1]

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

We consider a modification of the well known work function algorithm (WFA) for solving the on-line *k*-server problem. Our modified WFA is based on a moving window, i. e. on the approximate work function that takes into account only a fixed number of most recent on-line requests. The main motivation for using a moving window is to gain control over the prohibitive computational complexity imposed by the original algorithm. Experimental results are presented, where the performance of the modified WFA has been compared vs. the original WFA.

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

1. Introduction

In this paper we deal with *on-line problems*, and corresponding *on-line algorithms* [4]. In an on-line problem, input data is supplied incrementally, and output is expected to be produced in the same fashion. A typical on-line algorithm works in steps: in the i-th step it receives a *request* (unit of input) r_i, and makes a *decision* (unit of output) d_i that determines how r_i should be served. Each d_i produces a cost $c_i \geq 0$. Decision making is based only on some of the previously seen requests r_1, r_2, ..., r_{i-1}, r_i, thus without any knowledge about the future requests r_{i+1}, r_{i+2}, ... The objective of the algorithm is not only to serve requests, but also to *reduce the total cost* $\sum_i c_i$.

An on-line algorithm ALG can be viewed as an approximation algorithm of a certain sort. Namely, ALG attempts to approximate the performance of the corresponding optimal off-line algorithm OPT, which knows the whole input in advance and deals with requests as they arrive at minimum total cost. Vaguely speaking, ALG is said to be *competitive* if there exists a theoretical evidence that its performance is "close" to that of OPT on each input. More precisely [8], let $\sigma = (r_1, r_2, \ldots, r_n)$ be a sequence of requests. Denote with $C_{\mathrm{ALG}}(\sigma)$ the total cost incurred by ALG on σ, and with $C_{\mathrm{OPT}}(\sigma)$ the minimum total cost on σ. For a chosen constant p, we say that ALG is p-competitive if there exists another constant q such that on every σ it holds: $C_{\mathrm{ALG}}(\sigma) \leq p \cdot C_{\mathrm{OPT}}(\sigma) + q$.

In this paper we concentrate on one particular on-line problem called the *k-server problem* [6], which is important because many other on-line problems can be interpreted as its special cases. There are various algorithms for solving the *k*-server problem found in literature, but most of them are known to be non-competitive. The only competitive algorithm is the so-called *work function algorithm* (WFA) [5]. In spite of its theoretical importance, the WFA is seldom used in practice due to its prohibitive and ever-increasing computational complexity.

The aim of this paper is to propose a simple modification of the original WFA, which is based on using a *moving window*. In contrast to the original WFA, which makes new decisions by considering all previous requests, the modified WFA should take into account only a fixed

number of most recent requests. We expect that with such modification the computational complexity of the algorithm will decrease, while, at the same time, the performance in terms of the incurred total cost will not degrade too much.

The paper is organized as follows. Section 2 gives preliminaries about the k-server problem and the corresponding algorithms. Section 3 describes how the modified WFA differs from the original WFA and how it can be implemented. Section 4 reports on experiments, where the performance of the modified WFA has been measured in comparison with the original WFA and some other algorithms. Section 5 offers a conclusion.

2. The k-server Problem

In the *k-server problem* [6] we have to decide how k mobile servers will serve a sequence of requests. More precisely, we have k servers each of which occupies a location (point) in a fixed metric space M consisting of altogether m locations. Repeatedly, a request r_i at some location $x \in M$ appears. Each request must be served before the next request appears. To serve the request at x, the algorithm must move a server to x unless it already has a server at that location. Whenever the algorithm moves a server from location a to location b, it incurs a cost equal to the distance between a and b in M. The goal is to minimize the total distance moved by all servers.

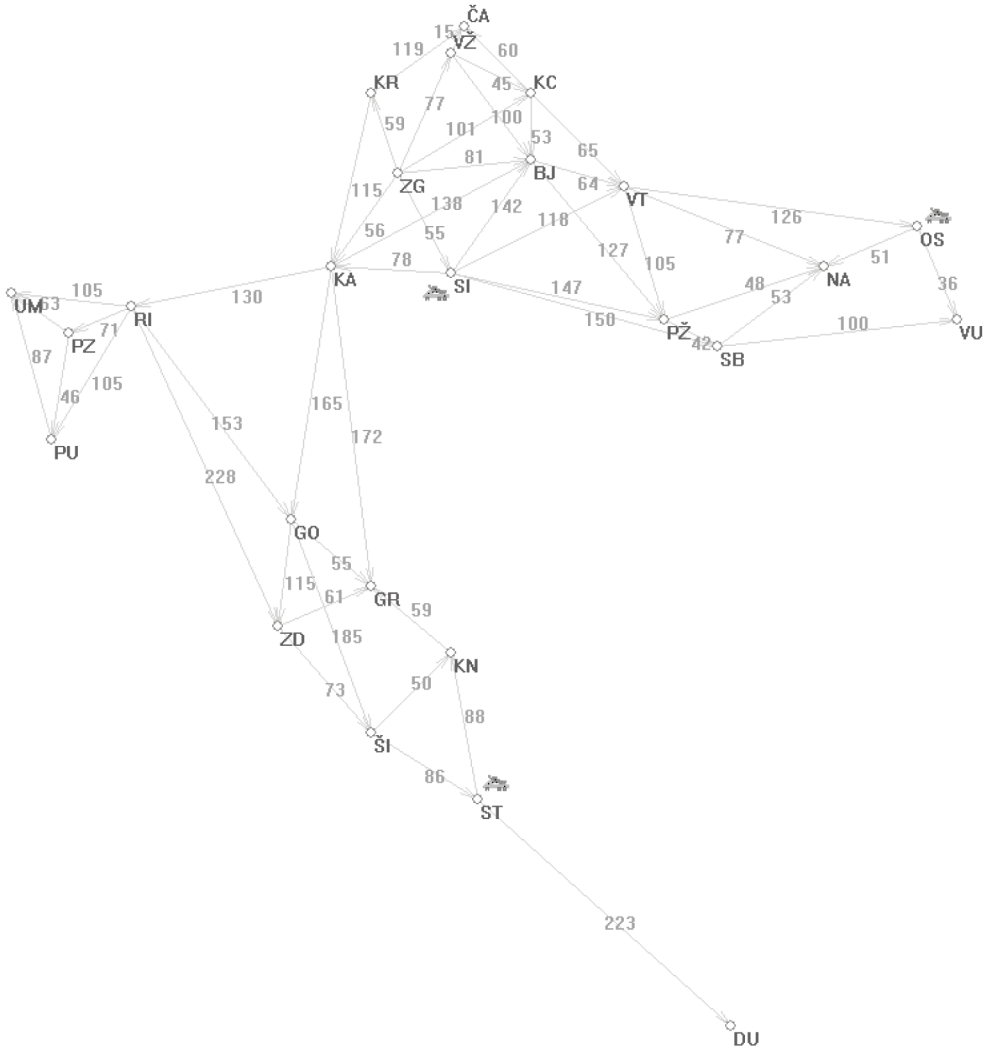

Figure 1. A k-server problem instance.

As an example of the k-server problem, let us consider the set M of Croatian cities shown in Figure 1 with road lengths given. The distance between any two cities is determined as the length of the shortest route over the available roads. Suppose that $k = 3$ different hail-defending rocket systems are initially located at OS, SI and ST. If the next hail alarm appears for instance in KA, then the hail-defending on-line algorithm has to decide which of the three rocket systems should be moved to KA. Seemingly the cheapest solution would be to move the nearest system from SI. But such a choice could be wrong if, for instance, all forthcoming requests would appear in SI, KA and OS and none in ST.

The simplest on-line algorithm for solving the k-server problem is the *greedy algorithm* (GREEDY) [4]. It serves the current request in the cheapest possible way, by ignoring history altogether. Thus GREEDY sends the nearest server to the requested location.

A slightly more sophisticated solution is the *balanced algorithm* (BALANCE) [6], which attempts to keep the total distance moved by various servers roughly equal. Consequently, BALANCE employs the server whose cumulative distance traveled so far plus the distance to the requested location is minimal.

The most celebrated solution to the k-server problem is the *work function algorithm* (WFA) [5]. To serve the request r_i, WFA switches from the current server configuration S_{i-1} to a new configuration S_i, obtained from S_{i-1} by moving one server into the requested location (if necessary). Among k possibilities (any of k servers could be moved) S_i is chosen so that it minimizes the so-called *work function* (WF). More precisely, S_i is chosen so that $C_{\mathrm{OPT}}(r_1, r_2, \ldots, r_i, S_i) + c(S_{i-1}, S_i)$ becomes minimal. The WF is defined here as a sum of two parts, the first one being the optimal cost of serving in turn $r_1, r_2, \ldots, r_i$ and ending up in S_i, the second one being the distance traveled by a server to switch from S_{i-1} to S_i.

There are many interesting theoretical results regarding on-line algorithms for the k-server problem. For instance, it can be proved [6] that any hypothetical p-competitive algorithm for the k-server problem must have $p \geq k$. Also, it is easy to check [4] that both GREEDY and BALANCE are not competitive, i.e. they have no bounded p. Finally, it has been proved in [6] that the WFA is $(2k-1)$-competitive. It is believed that the WFA is in fact k-competitive (thus achieving the best possible p), but this hypothesis has not been proved, except for some special cases [1].

3. The Modified WFA

In order to implement the WFA, we must also consider the off-line version of the k-server problem, i.e. the version where the whole sequence of requests is known in advance. It is important to note that the optimal off-line algorithm OPT can be implemented relatively easily by network flow techniques [2]. Namely, according to [3], finding the optimal strategy to serve a sequence of n requests by k servers can be reduced to computing the min-cost-max-flow on a suitably constructed network with $2n+k+2$ nodes. The details of this construction are available in [7].

According to the definition from Section 2, the decision taken in the i-th step of the WFA depends on the whole list of requests r_1, r_2, ..., r_{i-1}, r_i, as well as on the initial server configuration S_0. Also, we can see that the i-th step consists of k optimization problem instances, plus some simple arithmetics. Thus there is a possibility to implement the WFA by using the above mentioned network flow techniques.

It is true, however, that the optimization problems within the WFA are not quite equivalent to off-line problems, namely there is an additional constraint regarding the final configuration of servers. Still, we can demonstrate that the construction from [3,7] can be used after a slight modification. More precisely, we can show that the i-th step of the WFA can be reduced to k min-cost-max-flow problems, each on a network with $2i + 2k + 2$ nodes. The details of our modified construction cannot be given here due to the page limit, so they will be presented in a forthcoming paper.

Now it becomes obvious why the original WFA is not suitable for practical purposes. Namely, the algorithm is fairly complex since each of its steps involves k min-cost-max-flow problems. Even worse, the complexity of the i-th step grows with i since each of the involved

networks consists of $2i + 2k + 2$ nodes. Consequently, the algorithm gradually slows down until it becomes intolerably slow.

To overcome the described problems with the original WFA, let us now propose our modification. The modified WFA, denoted more precisely as the w-WFA, is based on the idea that the sequence of previous requests and configurations should be examined through a moving window of size w. More precisely, in its i-th step the w-WFA acts as if r_{i-w+1}, r_{i-w+2}, ..., r_{i-1}, r_i was the whole sequence of previous requests, and as if S_{i-w} was the initial configuration of servers. Obviously, the i-th step of the w-WFA can be implemented as k min-cost-max-flow problems, each on a network with $2w + 2k + 2$ nodes.

Note that each step of the w-WFA has roughly the same computational complexity. Thus the algorithm does not slow down any more. Moreover, the complexity of one step can be controlled by the window size w, i.e. smaller w means faster response. On the other hand, one can hope that with a sufficiently large w the w-WFA would closely approximate the behaviour of the original WFA. It must be admitted, however, that we cannot guarantee any more that the w-WFA is still a competitive algorithm.

4. Experimental Results

In order to obtain experimental results, we have developed a computer program that implements the w-WFA for any given w. Our implementation is based on reducing the w-WFA to network flows, as described in Section 3. The involved min-cost-max-flow problems are solved by a generic algorithm, which first determines a maximal flow and then iteratively improves that flow by finding cost-reducing cycles [2].

To allow comparison, we have also implemented some other algorithms for solving the k-server problem, such as GREEDY, BALANCE, and the original WFA. Also, we have made a program that implements the corresponding optimal off-line algorithm OPT. Note that GREEDY is equivalent to the 1-WFA, and the original WFA to the ∞-WFA.

The algorithms have been tested on 5 k-server problem instances, which are all based on locations and distances from Figure 1. An instance is characterized by its number of chosen locations m (ranging from 7 to 25), number of servers k (being 3 or 6), and number of consecutive requests n (100 or 200). In each instance, the initial server configuration is specified by hand, while the sequence of requests is produced automatically by a random number generator. The distribution of requests among locations is uniform in 2 instances and non-uniform in 3 instances.

The results of our experimental evaluation are given in Table 1. Each row in the table corresponds to a particular algorithm, and each column to a particular problem instance. Each entry records the performance of the corresponding algorithm on the corresponding problem instance. Performance is expressed in terms of the incurred total cost.

From Table 1 we can see that the w-WFA indeed achieves the same performance as the original WFA if w is large enough. For our problem instances, the equivalence between the w-WFA and the original WFA is reached with w between 16 and 30, as denoted by boldface printing.

According to Table 1, simpler versions of w-WFA can sometimes (apparently by pure chance) outperform more complex versions. For example, on problem instance #4, the 3-WFA is better than any other w-WFA, even the original WFA. Such anomalies are to be expected when the distribution of requests among locations is uniform, i. e. in situations when history does not count very much.

Table 1 also shows that the WFA (or the w-WFA respectively) performs better than BALANCE on all problem instances, and better than GREEDY on 4 out of 5 instances. Moreover, the WFA is superior to the other algorithms on all 3 instances with non-uniform distribution of requests.

The experiments have also confirmed that the algorithms differ considerably in their computational complexity. For example, to serve all 200 requests of problem instance #5, GREEDY needs on our PC less than 1 ms, BALANCE approximately 1 ms, the 2-WFA about 20 ms, the 18-WFA already 246 ms, and the original WFA even 115508 ms. A more detailed report on running times has been skipped here due to the page limit, and it will be presented in a forthcoming paper.

problem instance	#1	#2	#3	#4	#5
number of chosen locations m	7	22	25	25	25
number of servers k	3	3	3	6	3
number of consecutive requests n	100	100	100	100	200
uniform distribution of requests?	yes	no	no	yes	no
OPT	5647	5801	6125	6498	6200
BALANCE	9559	8455	8825	10898	10218
GREEDY	6752	8666	9049	9634	8028
2-WFA	6752	8918	9049	10330	8174
3-WFA	6862	9095	9049	9993	8336
4-WFA	6862	8941	9049	10019	7916
5-WFA	6512	8748	8922	10469	7858
6-WFA	6512	8646	8576	10176	7858
7-WFA	6842	8646	8882	10310	7924
8-WFA	7280	8646	8882	10463	7924
9-WFA	7502	8561	8646	10528	7924
10-WFA	7502	8561	8649	10376	7924
11-WFA	7502	8561	8649	10636	7924
12-WFA	6888	8299	8649	10412	7924
13-WFA	6558	8508	8649	10412	7924
14-WFA	6558	8508	8649	11367	7924
15-WFA	6558	8512	8649	10619	7924
16-WFA	6558	8508	**8747**	10852	7924
17-WFA	6558	8736	8747	10667	7924
18-WFA	6558	8743	8747	10571	**7992**
19-WFA	6796	8583	8747	10588	7992
20-WFA	6796	8583	8747	10620	7992
21-WFA	6796	8583	8747	10648	7992
22-WFA	6796	8583	8747	10742	7992
23-WFA	6888	**8062**	8747	10781	7992
24-WFA	6888	8062	8747	10781	7992
25-WFA	6888	8062	8747	10781	7992
26-WFA	6888	8062	8747	10781	7992
27-WFA	6888	8062	8747	10781	7992
28-WFA	**6558**	8062	8747	10781	7992
29-WFA	6558	8062	8747	10781	7992
30-WFA	6558	8062	8747	**10680**	7992
31-WFA	6558	8062	8747	10680	7992
32-WFA	6558	8062	8747	10680	7992
33-WFA	6558	8062	8747	10680	7992
34-WFA	6558	8062	8747	10680	7992
35-WFA	6558	8062	8747	10680	7992
36-WFA	6558	8062	8747	10680	7992
37-WFA	6558	8062	8747	10680	7992
38-WFA	6558	8062	8747	10680	7992
39-WFA	6558	8062	8747	10680	7992
40-WFA	6558	8062	8747	10680	7992
original WFA	6558	8062	8747	10680	7992

Table 1. Performance (total cost) of various algorithms on chosen k-server problem instances.

5. Conclusion

In this paper we have proposed a modified work function algorithm (WFA) for solving the on-line k-server problem, which is based on a moving window. Contrary to the standard WFA, our modified WFA is suitable for practical purposes since its computational complexity is bounded and can be controlled by the window size. The complexity of the modified WFA is still large compared to simple heuristics, such as the greedy or the balanced algorithm. However, this additional computational effort can be tolerated if it assures better performance, i. e. smaller total cost of responding to requests.

The presented experimental results clearly indicate that with a reasonably large window the modified WFA achieves the same performance as the original WFA, thus gaining all virtues of competitive computing. Consequently, the modified WFA usually outperforms the mentioned simple heuristics. The advantages of the WFA are more visible on problem instances with non-uniform distribution of requests.

Our plan is to develop a distributed implementation of the modified WFA. By employing more processors it should be possible to speed up the algorithm in order to meet strict response time requirements of on-line computation.

References

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

[2] M. S. BAZARAA, J. J. JARVIS, H. D. SHERALI, *Linear Programming and Network Flows*. Third edition, Wiley-Interscience, 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), 172–181.

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

[5] E. KOUTSOUPIAS, C. PAPADIMITROU, On the k-server conjecture. In: *Proceedings of the 26-th Annual ACM Symposium on Theory of Computing – Montreal Quebec, Canada, May 23-25, 1994* (F.T. Leighton, M. Goodrich, Eds). ACM Press, New York, (1994), 507–511.

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

[7] T. RUDEC, The k-server Problem. MSc Thesis (in Croatian), Department of Mathematics, University of Zagreb, 2001.

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

Received: June, 2007
Accepted: December, 2007

Contact addresses:

Alfonzo Baumgartner
Faculty of Electrical Engineering
University of Osijek
Kneza Trpimira 2b
31 000 Osijek, Croatia
e-mail: Alfonzo.Baumgartner@etfos.hr

Robert Manger
Department of Mathematics
University of Zagreb
Bijenička cesta 30
10 000 Zagreb, Croatia
e-mail: Robert.Manger@math.hr

Željko Hocenski
Faculty of Electrical Engineering
University of Osijek
Kneza Trpimira 2b
31 000 Osijek, Croatia
e-mail: Zeljko.Hocenski@etfos.hr

ALFONZO BAUMGARTNER is a research assistant and a PhD student at the Faculty of Electrical Engineering, Josip Juraj Strossmayer University in Osijek, Croatia. His current research interests include on-line problems, parallel algorithms applied to combinatorial optimization problems and other algorithms which use efficient data structures. He has published 5 papers in international scientific journals or 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: parallel and distributed algorithms, combinatorial optimization, and soft computing. He has published 18 papers in international scientific journals, over 20 scientific papers in conference proceedings, 10 professional papers, and 3 course materials. Dr Manger is a member of the Croatian Mathematical Society, Croatian Society for Operations Research and IEEE Computer Society.

ŽELJKO HOCENSKI is a professor at the Faculty of Electrical Engineering, Josip Juraj Strossmayer University in Osijek, Croatia. He is currently the head of a research project dealing with distributive and dependable embedded computer systems. His research interests include fault tolerant computing, parallel computing, system design, computer vision diagnostics, image processing and software engineering. He has published over 40 papers in international scientific journals or conference proceedings, a number of professional papers, and 2 course materials.