ELSEVIER

Operations Research Letters 27 (2000) 93–96

operations research letters

www.elsevier.com/locate/dsw

Monge strikes again: optimal placement of web proxies in the internet [☆]

Gerhard J. Woeginger

Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria

Received 1 June 1999; received in revised form 1 May 2000

Abstract

In a recent paper (Proceedings of IFIP'98), Li et al. study the problem of placing m web proxies in the internet under a given traffic pattern. They consider the special case of a linear net topology with n nodes. Their goal is to minimize the overall latency of accessing the target web server subject to the network resources and traffic pattern. They show how this problem can be solved in $O(n^2m)$ time. In this short note, we observe that one of the underlying cost functions in this problem carries a Monge structure. By exploiting this structure and by applying some well-known results from the literature, we get a faster algorithm with time complexity $O(nm)$. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Optimization; Dynamic programming; Monge property; Internet; Caching; Web proxy; Web server

1. Introduction

The poor performance and the long response times on the internet are due to many reasons, like congestion during peak times, links with inadequate bandwidth, long propagation delays, etc. *Caching* is one method for dealing with some of these problems. The main idea in caching is to keep retrieved documents close to the clients and thus to bring down the access latency. Essentially, caching can be done either at the client's web browser, or at the web server itself, or at a web proxy [10]. A *web proxy* is an intermediate server acting as a caching agent between clients and web servers. There has been considerable work done on various aspects of web proxies, see e.g. [2,5]. The effectiveness of a proxy is primarily determined by its *locality* which in turn depends on a number of factors such as access patterns and configurations. Placing a web proxy at the 'wrong' place is not only costly, but also does little to improve the performance.

Li et al. [7] study the problem of minimizing the overall latency of accessing the target web server. They consider the special case of a linear net topology. Such a net topology is a path with nodes $V = \{v_0, v_1, \ldots, v_n\}$ and edges $[v_{j-1}, v_j]$ for $1 \leqslant j \leqslant n$. For $1 \leqslant j \leqslant n$, the weight w_j of v_j represents the traffic traversing this node. For $1 \leqslant j \leqslant n$, the length of edge $[v_{j-1}, v_j]$ is measured by a non-negative real value d_j which can be interpreted as either latency, or link cost,

[☆] Supported by the START program Y43-MAT of the Austrian Ministry of Science.

E-mail address: gwoegi@opt.math.tu-graz.ac.at (G.J. Woeginger).

or hop count, etc. The distance $d(v_i, v_j)$ between two nodes v_i and v_j is the total length of all edges on the path between v_i and v_j. There is a target web server at node v_0 whose data should be accessed by every node v_j. If there is a proxy located between v_0 and v_j, then v_j only needs to access the data cached at the proxy; this reduces the traffic on the net between v_0 and the proxy and thus also reduces the overall accessing latency.

Let $\mathscr{P} = \{v_{p_1}, v_{p_2}, \ldots, v_{p_m}\}$ with $1 \leqslant p_1 < p_2 < \cdots < p_m \leqslant n$ be a set of m proxies located on this linear net. Then every node v_j with $j < p_1$ accesses the data directly at the target web server at v_0 (and thus does not use any of the proxies); we set $\mathscr{P}(v_j) = v_0$. For $1 \leqslant k \leqslant m-1$, the nodes v_j with $p_k \leqslant j < p_{k+1}$ access the data of proxy v_{p_k}, and we set $\mathscr{P}(v_j) = v_{p_k}$. Finally, the nodes v_j with $j \geqslant p_m$ access proxy v_{p_m}, and we set $\mathscr{P}(v_j) = v_{p_k}$. Then the *latency* caused by node v_j equals $w_{jd}(v_j, \mathscr{P}(v_j))$, and the overall accessing latency with proxy set $\mathscr{P}$ is

$$\sum_{j=1}^{n} w_j d(v_j, \mathscr{P}(v_j)). \tag{1}$$

The goal is to find in $O(n^2 m)$ time a set $\mathscr{P}$ of m proxies that minimizes the objective value in (1). In the language of facility location, this problem is the asymmetric k-median problem on a directed path. We refer the reader to Li et al. [7,8] for more information on this problem and for a discussion of some of its more complex variants. The paper [7] finds in $O(n^2 m)$ time the best possible proxy set; this is done by modelling the problem as a dynamic program.

Contribution of this note. We reformulate the dynamic program of Li et al. [7], and we observe that one of the underlying cost functions carries a it Monge structure. Such Monge structures show up in a wide variety of applications like transportation problems, flow problems, search problems, Travelling salesman variants, economic lot sizing problems, geometric problems, etc. Usually, this additional structure leads to polynomial-time solution algorithms or at least helps to speed up algorithms. For more information on Monge structures, we refer the reader to the survey article by Burkard et al. [3]. In this note, we exploit this Monge structure in the proxy placement problem. By applying some well-known results from the literature, we implement the dynamic programming algorithm within $O(nm)$ time. This is an improvement of $O(n)$ over the time complexity of Li et al. [7].

This note is organized as follows. In Section 2 we summarize some results from the literature on searching and optimizing under Monge structures. In Section 3 we apply the results of Section 2 to the proxy placement problem and thereby derive a solution algorithm with $O(nm)$ running time.

2. Dynamic programming under Monge structures

In this section, we recapitulate some results of Aggarwal et al. [1] on a problem that comes up as a subproblem in certain dynamic programming formulations: Let $a(i,j)$ be a real-valued function for integers $0 \leqslant j \leqslant i \leqslant n$ that fulfills the following two conditions.

(C1) For $0 \leqslant i \leqslant j \leqslant n$, the value $a(i,j)$ can be computed in constant time.
(C2) For $0 \leqslant i \leqslant r \leqslant j \leqslant s \leqslant n$, the inequality $a(i,j) + a(r,s) \leqslant a(i,s) + a(r,j)$ holds.

Condition (C1) essentially states that we can access the function $a(i,j)$ via a very fast oracle. Condition (C2) essentially states that the $n \times n$ matrix A with entries $a(i,j)$ above the main diagonal is an *upper triangular Monge* matrix (the entries below the main diagonal of A are irrelevant for this Monge property); cf. [3] for more information on this concept.

Proposition 2.1 (Aggarwal et. al [1]). *If conditions* (C1) *and* (C2) *are fulfilled, then all values* $E[j]$, $j = 1 \ldots n$, *that are specified by*

$$E[j] = \min\{a(i,j) : 1 \leqslant i < j\} \tag{2}$$

can be computed in $O(n)$ *overall time.*

Proposition 2.2. *Let m and n be two integers with $m \leqslant n$. Let $a(i,j)$ be a real-valued function for integers $0 \leqslant j \leqslant i \leqslant n$ that satisfies* (C1) *and* (C2). *Let* $F[j,1]$ *with* $1 \leqslant j \leqslant n$ *be given real numbers. Then all values* $F[j,k]$ *with* $1 \leqslant j \leqslant n$ *and* $2 \leqslant k \leqslant m$ *that are specified by*

$$F[j,k] = \min\{F[i,k-1] + a(i,j) : 1 \leqslant i < j\} \tag{3}$$

can be computed in $O(nm)$ *overall time.*

Proof. The computation goes through $m-1$ phases. In the kth phase ($k=2\dots m$) we compute all values $F[j,k]$ from the values $F[j,k-1]$ in O(n) overall time. Clearly, this will prove the theorem. Indeed, let $k\geqslant 2$ and assume that we already have computed all values $F[j,k-1]$ with $1\leqslant j\leqslant n$. For integers $0\leqslant j\leqslant i\leqslant n$ define the real-valued auxiliary function

$$b_k(i,j):=F[i,k-1]+a(i,j).$$

With this, the recursion in (3) becomes a special case of the recursion in (2). Moreover, since the values $F[i,k-1]$ are already known and since $a(i,j)$ satisifies condition (C1), also the auxiliary function $b_k(i,j)$ can be evaluated in constant time. Finally, it is easily verified that if $a(i,j)$ fulfills the Monge condition (C2), then also $b_k(i,j)$ does fulfill (C2). Summarizing, $b_k(i,j)$ fulfills conditions (C1) and (C2), and according to Proposition 2.1 we can compute all values $F[j,k]$ with $1\leqslant j\leqslant n$ in O(n) time. □

More general (and much more sophisticated) results on how to exploit Monge structures to speed up dynamic programming can be found in the work of Wilber [9], Eppstein [4], and Larmore and Schieber [6].

3. A fast algorithm for the optimal placement of proxies

In this section, we show that the proxy placement problem can be solved in O(nm) time by applying Proposition 2.2 to it. To simplify the presentation, we introduce the numbers $d_0=0$ and $w_0=0$. For integers $0\leqslant i\leqslant j\leqslant n$ we define a function value $\tilde{a}(i,j)$ that measures the cost incurred on the segment between two adjacent proxies at v_i and v_j in a partial solution:

$$\tilde{a}(i,j)=\sum_{\ell=i+1}^{j-1} w_\ell d(v_\ell,v_i)=\sum_{\ell=i+1}^{j-1}\sum_{q=i+1}^{\ell} w_\ell d_q. \tag{4}$$

(Recall that $d_q=d(v_{q-1},v_q)$ is the length of the edge $[v_{q-1},v_q]$.) As usual, we assume that an empty sum takes value 0. Next, we will derive several properties of this function $\tilde{a}(i,j)$. To this end, we define the following auxiliary values for $0\leqslant j\leqslant n$.

$$D[j]=\sum_{q=0}^{j} d_q,\qquad W[j]=\sum_{\ell=j}^{n} w_\ell,$$

$$X[j]=\sum_{q=0}^{j}\sum_{\ell=q}^{n} w_\ell d_q,\qquad Y[j]=\sum_{\ell=j}^{n}\sum_{q=0}^{\ell} w_\ell d_q,$$

Lemma 3.1. *All the auxiliary values $D[j]$, $W[j]$, $X[j]$, and $Y[j]$ with $0\leqslant j\leqslant n$ can be computed in* O(n) *overall time.*

Proof. Observe that $D[0]=0$ and $D[j]=D[j-1]+d_j$ holds for $j>0$. Similarly, $W[n]=w_n$ and $W[j]=W[j+1]+w_j$ holds for $j<n$. Hence, all values $D[j]$ and $W[j]$ can be computed in O(n) time by going from $j=0$ up to $j=n$, respectively, by going from $j=n$ down to $j=0$. Next, observe that $X[0]=0$, that $X[j]=X[j-1]+d_jW[j]$ for $j>0$, that $Y[n]=w_nD[n]$, and that $Y[j]=Y[j+1]+w_jD[j]$ for $j<n$. Hence, once we have determined the values $D[j]$ and $W[j]$, also all the values $X[j]$ and $Y[j]$ can be computed in O(n) time. □

Lemma 3.2. *For all integers $0\leqslant i\leqslant j\leqslant n$ function $\tilde{a}(i,j)$ specified in* (4) *satisfies*

$$\tilde{a}(i,j)=Y[0]-Y[j]-X[i]+D[i]\cdot W[j] \tag{5}$$

Proof. By plugging in the definitions of $D[j]$, $W[j]$, $X[j]$, and $Y[j]$ we get that the value of $Y[0]-Y[j]-X[i]+D[i]\cdot W[j]$ equals

$$\sum_{\ell=0}^{n}\sum_{q=0}^{\ell} w_\ell d_q-\sum_{\ell=j}^{n}\sum_{q=0}^{\ell} w_\ell d_q-\sum_{q=0}^{i}\sum_{\ell=q}^{n} w_\ell d_q$$
$$+\sum_{q=0}^{i}\sum_{\ell=j}^{n} w_\ell d_q$$
$$=\sum_{\ell=0}^{j-1}\sum_{q=0}^{\ell} w_\ell d_q-\sum_{q=0}^{i}\sum_{\ell=q}^{j-1} w_\ell d_q$$
$$=\sum_{q=0}^{j-1}\sum_{\ell=q}^{j-1} w_\ell d_q-\sum_{q=0}^{i}\sum_{\ell=q}^{j-1} w_\ell d_q$$
$$=\sum_{q=i+1}^{j-1}\sum_{\ell=q}^{j-1} w_\ell d_q.$$

To complete the argument, we observe that $\sum_{q=i+1}^{j-1}\sum_{\ell=q}^{j-1} w_\ell d_q$ equals $\tilde{a}(i,j)$ as defined in (4). □

Lemma 3.3. *After a preprocessing phase that takes* O(n) *time, we can compute for any i and j with* $0 \leqslant i \leqslant j \leqslant n$ *the value* $\tilde{a}(i,j)$ *in constant time. Hence, after this preprocessing phase function* $\tilde{a}(i,j)$ *fulfills condition* (C1). *Moreover, function* $\tilde{a}(i,j)$ *fulfills condition* (C2).

Proof. The preprocessing phase consists in computing the values $D[j]$, $W[j]$, $X[j]$, and $Y[j]$ as described in Lemma 3.1. By Lemma 3.2 we then can compute $\tilde{a}(i,j)$ in constant time. This proves the first part of the lemma, and it remains to prove that $\tilde{a}(i,j)$ fulfills condition (C2). Let $0 \leqslant i \leqslant r \leqslant j \leqslant s \leqslant n$ and use (5) to get

$$\begin{aligned}\tilde{a}(i,s) + \tilde{a}(r,j) - \tilde{a}(i,j) - \tilde{a}(r,s) \\ &= D[i]W[s] + D[r]W[j] - D[i]W[j] - D[r]W[s] \\ &= (D[r] - D[i])(W[j] - W[s]).\end{aligned}$$

Observe that since the w_ℓ and d_q are non-negative numbers, $D[r] \geqslant D[i]$ and $W[j] \geqslant W[s]$ holds. Hence, the product $(D[r]-D[i])(W[j]-W[s])$ is non-negative and condition (C2) indeed is fulfilled. □

Theorem 3.4. *The problem of placing m web proxies in a linear net topology with n nodes so as to minimize the overall accessing latency in* (1) *can be solved in* O(nm) *time.*

Proof. For $1 \leqslant j \leqslant n$, we denote $V_j = \{v_1, \ldots, v_j\}$. For $1 \leqslant j \leqslant n$ and $1 \leqslant k \leqslant m$, we denote by $F[j,k]$ the minimum total latency caused by the nodes in V_j (disregarding the nodes in $V - V_j$), given that (i) at most k of the nodes in V_j are proxies, and that (ii) node v_j itself is a proxy. Then for $1 \leqslant j \leqslant n$, $F[j,1] = \sum_{\ell=1}^{j-1} w_\ell d(v_\ell, v_0) = \tilde{a}(0,j)$ holds. For $1 \leqslant k \leqslant m$, $F[1,k] = 0$. Moreover for $2 \leqslant j \leqslant n$ and $2 \leqslant k \leqslant m$ we have

$$F[j,k] = \min\{F[i,k-1] + \tilde{a}(i,j) : 1 \leqslant i < j\}. \quad (6)$$

The correctness of (6) is easy to verify: Consider a proxy configuration with k proxies on V_j that satisfies (i) and (ii). Let i denote the maximal index $1 \leqslant i \leqslant j-1$ in this configuration for which node v_i has a proxy. Then the latency caused by the nodes $v_{i+1}, \ldots, v_{j-1}$ (which all access the data of proxy v_i) equals $\sum_{\ell=i+1}^{j-1} w_\ell d(v_\ell, v_i)$ which is exactly $\tilde{a}(i,j)$. By the definition of $F[\cdot,\cdot]$, the latency caused by the remaining nodes $v_1, \ldots, v_i$ with $k-1$ proxies is at least $F[i,k-1]$. Hence, the expression in (6) simply selects the proxy position v_i that minimizes the total latency.

The recursion in (6) is of the form in (3). We compute the values $D[j]$, $W[j]$, $X[j]$, and $Y[j]$ in O(n) time as described in Lemma 3.1. Then all initial values $F[j,1] = \tilde{a}(0,j)$ with $1 \leqslant j \leqslant n$ can be determined in O(n) time. By Lemma 3.3, the function $\tilde{a}(i,j)$ fulfills conditions (C1) and (C2). Consequently, by Proposition 2.2 we can determine all values $F[j,k]$ with $2 \leqslant j \leqslant n$ and $1 \leqslant k \leqslant m$ in O(mn) overall time. In order to compute the optimal objective value, we consider the proxy node v_i with highest index in an optimal solution. Then in this optimal solution the nodes $v_1, \ldots, v_i$ cause a total latency of $F[i,m]$ and the nodes $v_{i+1}, \ldots, v_n$ cause a total latency $\tilde{a}(i,n)$. With this, the value $\min\{F[i,m] + \tilde{a}(i,n) : m \leqslant i \leqslant n\}$ constitutes the optimal objective value. □

References

[1] A. Aggarwal, M.M. Klawe, S. Moran, P. Shor, R. Wilber, Geometric applications of a matrix-searching algorithm, Algorithmica 2 (1987) 195–208.

[2] M.F. Arlitt, C.L. Williamson, Internet web servers: Workload characterization and performance implications, IEEE Trans. Networking 5 (1997) 631–645.

[3] R.E. Burkard, B. Klinz, R. Rudolf, Perspectives of Monge properties in optimization, Discrete Appl. Math. 70 (1996) 95–161.

[4] D. Eppstein, Sequence comparison with mixed convex and concave costs, J. Algorithms 11 (1990) 85–101.

[5] S. Glassman, A caching relay for the world wide web, Computer Networks and ISDN Systems 27 (1994) 165–173.

[6] L.L. Larmore, B. Schieber, On-line dynamic programming with applications to the prediction of RNA secondary structure, J. Algorithms 12 (1991) 490–515.

[7] B. Li, X. Deng, M. Golin, K. Sohraby, On the optimal placement of web proxies in the internet: Linear topology, Proceedings of the 8th IFIP Conference on High Performance Networking (HPN'98), Vienna, Austria, September 1998.

[8] B. Li, M. Golin, G. Italiano, X. Deng, K. Sohraby, On the optimal placement of web proxies in the internet, Proceedings of the IEEE Infocom Conference, March 1999.

[9] R. Wilber, The concave least-weight subsequence problem, J. Algorithms 9 (1988) 418–425.

[10] N. Yeager, R. McGrath, Web Server Technology. Morgan Kaufman, Los Attos, CA (1996).