Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 586146, 11 pages
http://dx.doi.org/10.1155/2014/586146

Research Article

Weighted Cache Location Problem with Identical Servers

Hongfa Wang and Wei Ding

Zhejiang University of Water Resources and Electric Power, Hangzhou, Zhejiang 310018, China

Correspondence should be addressed to Wei Ding; dingweicumt@163.com

Received 16 February 2014; Revised 20 June 2014; Accepted 4 August 2014; Published 24 August 2014

Academic Editor: Qi Cheng

This paper extends the well-known p-CLP with one server to p-CLP with $m \geq 2$ identical servers, denoted by (p, m)-CLP. We propose the closest server orienting protocol (CSOP), under which every client connects to the closest server to itself via a shortest route on given network. We abbreviate (p, m)-CLP under CSOP to (p, m)-CSOP CLP and investigate that (p, m)-CSOP CLP on a general network is equivalent to that on a forest and further to multiple CLPs on trees. The case of $m = 2$ is the focus of this paper. We first devise an improved $O(ph^2+n)$-time parallel exact algorithm for p-CLP on a tree and then present a parallel exact algorithm with at most $O((4/9)p^2n^2)$ time in the worst case for $(p, 2)$-CSOP CLP on a general network. Furthermore, we extend the idea of parallel algorithm to the cases of $m > 2$ to obtain a worst-case $O((4/9)(n-m)^2((m+p)^p/(p-1)!))$-time exact algorithm. At the end of the paper, we first give an example to illustrate our algorithms and then make a series of numerical experiments to compare the running times of our algorithms.

1. Introduction

Caching has become an important tool to improve the network performance efficiency, reducing delays to every client and alleviating the overload on the server [1–4]. Initially, a large amount of studies considered how to optimize cache performance [5–7], cache hierarchies [5], and cooperations among multiple web servers [8, 9]. Subsequently, how to locate caches or proxies optimally in networks to alleviate the server load became more popular [2, 10–13]. The most popular practice in the past is to place caches on the edges of networks, acting as the network browser and proxy or part of cache hierarchies [1, 3–5]. Later, Danzig et al. [2] discovered that the advantage of placing caches on the nodes of networks instead of on the edges of networks is to reduce overall network congestion greatly. In this paper, we only discuss how to place caches on the nodes of networks.

The focus of placing caches in networks is how to enhance the effect and efficiency of caching in networks as greatly as possible. This problem can be modeled as the *p-cache location problem* (abbreviated to p-CLP or CLP) or *p-proxy problem*. Both of their initial models can be reduced to the*p-median problem* [14, 15] essentially. Throughout this paper, we let n denote the number of network nodes, let m denote the number of network edges, let p denote the number of caches or proxies, let and h denote the height of tree. Later, Abrams et al. [7] investigated that almost all current cache products contain a transparent operation mode, called a *transparent en-route cache* (TERC). When using TERCs in networks with one server, all clients connect to server and caches are placed on the routes from clients to server. Heddaya and Mirdad [10] suggested making use of TERCs to balance load due to the manageability of TERC. Further, Krishnan et al. [11] proposed the cache location problem involving TERCs, studied the problem in several special networks, and presented polynomial time exact algorithms. In the rest of this paper, all of CLPs involve TERC.

The known algorithms for the *p-proxy problem* also apply to p-CLP. For a linear network, Li et al. studied the p-proxy problem and presented an $O(pn^2)$ time exact algorithm [12]. Later, Woeginger used the Monge property to obtain an improved algorithm with $O(pn)$ time complexity [16]. For a tree network, Li et al. devised an $O(p^2n^3)$-time exact dynamic programming algorithm [13] and Chen et al. presented an improved $O(nph)$-time algorithm [17]. For a general tree of rings network, Chen et al. showed an $O(pn^2)$ time exact algorithm [18]. Moreover, a variety of objectives have been considered, such as the overall time, cost, and hop count. Du [19] and Jia et al. [20] studied the proxy problem with read-write operations. In [21], Liu and Yang considered the delay-constrained proxy problem. Given a general network, provided that all clients are *intelligent* and connect to the server via a *shortest route* on the network, we claim that p-CLP in this situation is equivalent to p-CLP on the tree and thus is solvable in polynomial time [11, 13, 17] since all the shortest routes between clients and server form a shortest-paths tree rooted at the server node.

All past works on CLP considered a single server. This paper is the first one to study CLP with multiple *identical* servers. In this model, every client connects to some server via a route and all caches lie on such routes. We further suppose that every client is *intelligent*; that is, it connects to the closest server to itself via a shortest route. This produces the *closest server orienting protocol* (abbreviated to CSOP). Under CSOP, all the shortest routes connecting clients and server form a *forest*, each component of which is rooted at a server node. Therefore, the *CLP with multiple identical servers under CSOP* on a general network is equivalent to that on a forest.

We abbreviate p-CLP with m identical servers to (p,m)-CLP and further (p,m)-CLP under CSOP to (p,m)-CSOP CLP. In this paper, we first propose an improved parallel exact algorithm for p-CLP on a tree, which reduces $O(nph)$ time of algorithm in [17] to $O(ph^2 + n)$. Based on the improved algorithm, we design a parallel exact algorithm for $(p,2)$-CSOP CLP on a general network, which takes $O(p^2 \max\{h_1^2, h_2^2\} + (8/3)np)$ time and at most $O((4/9)p^2n^2)$ time in the worst case. Furthermore, we extend the algorithm idea to (p,m)-CSOP CLP on a general network and get a parallel exact algorithm, which takes $O((p \cdot \max_{1\le i\le m} h_i^2 + (8/3)n)\binom{m+p-1}{p})$ time and at most $O((4/9)(n-m)^2((m+p)^p/(p-1)!))$ time in the worst case.

The rest of this paper is organized as follows. In Section 2, we define notations used frequently and $(p,2)$-CSOP CLP formally. In Section 3, we make some fundamental preliminaries, develop an improved algorithm for p-CLP on a tree, and then devise a parallel exact algorithm for $(p,2)$-CSOP CLP based on the improved algorithm. In Section 4, we define (p,m)-CSOP CLP formally and devise an efficient parallel exact algorithm. In Section 5, we first give an example to illustrate our algorithm and then make a series of numerical experiments to compare the running time of our algorithm. In Section 6, we conclude this paper with some future research topics.

2. Problem Description

Let $G = (V,E,w,c)$ represent a communication network or computer network, where V is the *node set* and E is the *edge set*. Every node represents a processing or switching element and every edge represents a communication link [22]. Every node $z \in V$ has a weight $w(z) > 0$ representing the *demand amount* of z, and every edge $e \in E$ has a weight $c(e) > 0$ representing the *cost* per demand. For any pair of nodes x and y, we let $[x,y]$ denote the edge of G between x and y and let $\pi(x,y)$ denote the *shortest path* in G connecting x and y. Let $c[x,y]$ denote the cost of edge $[x,y]$, and $c(x,y)$ denote the cost of $\pi(x,y)$ which is equal to the sum of all the costs on edges of $\pi(x,y)$. So,

$$c(x,y) = \sum_{e\in\pi(x,y)} c(e). \tag{1}$$

Let s_1 and s_2 be two *identical* servers, which are allocated to a pair of nodes of G in advance. Let Γ denote the set of p cache locations. Suppose that CSOP works and each cache is a *duplicate* of server. Given any set $\Gamma \subset V \setminus \{s_1,s_2\}$, the cost of node z paying for its per demand depends on the locations of $\Gamma \cup \{s_1,s_2\}$ and is denoted by $c(z,\Gamma\cup\{s_1,s_2\})$. Thus, the cost of z paying for its overall demand is equal to $w(z)c(z,\Gamma\cup\{s_1,s_2\})$. Let $f(\Gamma)$ denote the total cost of all the nodes paying for their overall demand, that is,

$$f(\Gamma) = \sum_{z\in V} w(z)\, c(z,\Gamma\cup\{s_1,s_2\}). \tag{2}$$

The *p-cache location problem with two identical servers under CSOP* (i.e., $(p,2)$-CSOP CLP) aims to find p cache locations in G to minimize the total cost of all the nodes paying for their overall demand. In other words, the aim of $(p,2)$-CSOP CLP can be reduced to find an optimal set Γ^* from $V \setminus \{s_1,s_2\}$ to minimize the value of $f(\Gamma)$, that is,

$$f(\Gamma^*) = \min_{\Gamma\subset V\setminus\{s_1,s_2\}} f(\Gamma). \tag{3}$$

3. A Parallel Exact Algorithm for $(p,2)$-CSOP CLP

In the scenario of $(p,2)$-CSOP CLP, every client knows the location of the closest server to itself and connects to it via a shortest route. If its service request encounters the closest cache on the route, it will get information therein. Otherwise, it get information from the server. Therefore, $(p,2)$-CSOP CLP can be viewed as the combination of two CLPs when s_1 and s_2 are predesignated to two locations of network. One is CLP with s_1 as the server and the other is CLP with s_2 as the server.

3.1. Preliminaries. Once s_1 and s_2 are fixed at two predesignated locations of G, it is certain that some nodes of G are closer to s_1 and the other nodes are closer to s_2. Let $V(s_1)$ be the set of nodes that are closer to s_1 and $V(s_2)$ be the set of nodes that are closer to s_2. Thus, Lemma 1 follows immediately. Let $T(s_1)$ (resp. $T(s_2)$) denote the *single-source shortest paths tree* in G with s_1 (resp. s_2) as the origin spanning $V(s_1)$ (resp. $V(s_2)$). We can use Dijkstra's algorithm [23] to compute $T(s_1)$ and $T(s_2)$. We can transform $T(s_1)$ (resp. $T(s_2)$) into a rooted tree with s_1 (resp. s_2) as the root without loss of generality. Furthermore, let $T_x(s_1)$ denote the subtree of $T(s_1)$ rooted at x for any $x \in V(s_1)$ and let $T_y(s_2)$ denote the subtree of $T(s_2)$ rooted at y for any $y \in V(s_2)$.

Lemma 1. $V = V(s_1) \bigcup V(s_2)$.

Specifically, we let $\Pi[s_1, s_2]$ denote the unique path on tree between s_1 and s_2 when G is a tree graph. Let $N(s_1)$ (resp. $N(s_2)$) be the set of nodes on $\Pi[s_1, s_2]$ which are closer to s_1 (resp. s_2) and let $V_x(s_1, s_2)$ be the subset of nodes which reach s_1 and s_2 via x for any node x on $\Pi[s_1, s_2]$. We investigate that every node in $V_x(s_1, s_2)$, $x \in N(s_1)$ belongs to $V(s_1)$ and every node in $V_y(s_1, s_2)$, $y \in N(s_2)$ belongs to $V(s_2)$, and vice versa. So, Lemma 2 follows. By Lemmas 1 and 2, we can compute $T(s_1)$ and $T(s_2)$ by applying the *depth-first search* (DFS) procedure to the tree, which only takes a linear time.

Lemma 2. $V(s_1) = \bigcup_{x \in N(s_1)} V_x(s_1, s_2)$, $V(s_2) = \bigcup_{y \in N(s_2)} V_y(s_1, s_2)$.

By Lemma 1, we are sure that one cache is placed either in $V(s_1)$ or $V(s_2)$. The set of cache locations in $V(s_1)$ and $V(s_2)$ is denoted by $\Gamma(s_1)$ and $\Gamma(s_2)$, respectively; thus, $\Gamma = \Gamma(s_1) \cup \Gamma(s_2)$, $\Gamma(s_1) = \Gamma \cap V(s_1)$ and $\Gamma(s_2) = \Gamma \cap V(s_2)$. Under CSOP, every $z \in V(s_1)$ connects to s_1 and thus the cost of paying for its overall demand is equal to $w(z) \cdot c(z, \Gamma(s_1) \cup \{s_1\})$. Similarly, the cost of $z \in V(s_2)$ is equal to $w(z) \cdot c(z, \Gamma(s_2) \cup \{s_2\})$. We can further rewrite (2) as

$$\begin{aligned} f(\Gamma) &= f(\Gamma(s_1) \cup \Gamma(s_2)) \\ &= \sum_{z \in V(s_1)} w(z) \cdot c(z, \Gamma(s_1) \cup \{s_1\}) \\ &\quad + \sum_{z \in V(s_2)} w(z) \cdot c(z, \Gamma(s_2) \cup \{s_2\}). \end{aligned} \tag{4}$$

Let p_1 be the number of caches in $\Gamma(s_1)$ and let p_2 be the number of caches in $\Gamma(s_2)$. Obviously, $p_1 + p_2 = p$. A combination of $\Gamma(s_1)$ having p_1 caches and $\Gamma(s_2)$ having p_2 caches is called a *cache allocation scheme* (abbreviated to CAS), denoted as (p_1, p_2)-CAS where p_1 and p_2 cannot be exchanged. Clearly, $(p, 2)$-CSOP CLP contains $p + 1$ CASs in total. For any (p_1, p_2)-CAS, $(p, 2)$-CSOP CLP is composed of two subproblems, that is, p_1-CLP in $T(s_1)$ and p_2-CLP in $T(s_2)$. Let $C(G, p)$ denote the minimum cost of $(p, 2)$-CSOP CLP in G, and let $C(T(s_1), p_1)$ denote the minimum cost of p_1-CLP in $T(s_1)$ and let $C(T(s_2), p_2)$ denote the minimum cost of p_2-CLP in $T(s_2)$. The cost of $(p, 2)$-CSOP CLP in G for any given (p_1, p_2)-CAS is equal to the sum of $C(T(s_1), p_1)$ and $C(T(s_2), p_2)$, and further $C(G, p)$ results from the optimal (p_1, p_2)-CASs; that is,

$$C(G, p) = \min_{p_1 + p_2 = p} \{C(T(s_1), p_1) + C(T(s_2), p_2)\}. \tag{5}$$

3.2. Preprocessing. In this subsection, we give a new method of transforming an arbitrary rooted tree into a binary tree. Let T be a rooted tree. For any nonleaf node x of T, the subgraph of T is composed of the edges between x and all its children are called a *star* of T with center x, denoted by $star(x)$. Let $d(x)$ denote the number of children of x. We process x in the following way:

(i) if $d(x) = 1$, then we add a new child y' to x and set both $c[x, y']$ and $w(y')$ to zero;

(ii) if $d(x) = 2$, then we need no work;

(iii) if $d(x) = k$, $k \geq 3$ (let all children of x be $x_1, x_2, \ldots, x_k$), then we delete the edges $(x, x_1), (x, x_2), \ldots, (x, x_k)$, add two new nodes y' and y'' to x, and set $w(y'), c(x, y')$ and $w(y''), c(x, y'')$ to zero. For each $t = 1, 2, \ldots, \lceil k/2 \rceil$, we add new edge (y', x_t) and set $c(y', x_t)$ to $c(x, x_t)$. For each $t = \lceil k/2 \rceil + 1, \ldots, k$, we add new edge (y'', x_t) and set $c(y'', x_t)$ to $c(x, x_t)$.

We use the above way recursively to process every node of $T = (V, E, w, c)$ top-down to obtain a binary tree $b(T) = (V_b, E_b, w_b, c_b)$. This idea can be described as algorithm BINY. Our way improves that one proposed by Chen et al. [18]. Moreover, we will analyze the performance of BINY in the following while they provided no analysis of their algorithm [18].

Theorem 3. *For each x of T with $d(x) = k \geq 3$, the subtree of $b(T)$ derived from transforming $star(x)$ by* BINY *has a height of $O(\log k)$ and has $5k/3 - 2$ dummy nodes added in the worst case.*

Proof. The essence of BINY processing $star(x)$ is to bisect all the children of x recursively. At the final step, two dummy nodes are added if four nodes are bisected into two groups of two nodes and three dummy nodes are added if three nodes are bisected into one group of two nodes and one group of one node. So, BINY adds the most dummy nodes in the worst case of $k = 3 \cdot 2^t$. In this case, the subtree of $b(T)$ derived from transforming $star(x)$ by BINY has a height of $t + 2$ and the number of dummy nodes added is $2(1 - 2^t)/(1 - 2) + 3 \cdot 2^t = 5 \cdot 2^t - 2$. In fact, we investigate that the subtree of $b(T)$ derived from a star with k nodes satisfying that $2^{t+1} + 1 < k \leq 2^{t+2}$ has a height of $t + 2$. So, $\log k \leq t + 2 < \log k + 1$. Therefore, the subtree of $b(T)$ derived from $star(x)$ has a height of $O(\log k)$. The number of all the dummy nodes added by BINY in the worst case is $5 \cdot 2^t - 2 = 5 \cdot 2^{\log(k/3)} - 2 = 5k/3 - 2$. □

Theorem 4. BINY *can transform T with n nodes into $b(T)$ with a height of at most $2(n - 1)/3$, which takes $O(n)$ time and adds at most $(5/3)n - 11/3$ dummy nodes.*

Proof. Suppose that T has K stars and every $star(x_i)$, $1 \leq i \leq K$, has k_i children. Obviously, $\sum_{i=1}^{K} k_i = |E| = n - 1$. By Theorem 3, BINY adds $5k_i/3 - 2$ dummy nodes for every $star(x_i)$ in the worst case of $k_i = 3 \cdot 2^{t_i}$, $t_i \geq 0$, $\forall 1 \leq i \leq K$. So, the number of dummy nodes added by BINY is $\sum_{i=1}^{K} (5k_i/3 - 2) = (5/3) \sum_{i=1}^{K} k_i - 2K \leq (5/3)(n - 1) - 2 = (5/3)n - 11/3$.

Next, we discuss the height of $b(T)$. We construct a worst-case tree T^{Δ} consisting of K three-children stars lined one by one. In other words, for every star of T other than the bottom one, two children of the star are leaves of T and the other one is the center of the next star. Clearly, T^{Δ} satisfies that $3K = n - 1$. Since the subtree of $b(T)$ derived from transforming

```
Input: a binary tree $B$ rooted at $s$;
Output: an optimal solution of $p$-CLP on $B$ and $F[s][s][p]$;
Step 0. Use DFS based algorithm in [24] to traverse $B$, and record
          the parent node of every node $x$ of $B$;
Step 1. for $k$ from 1 up to $h$ do
          Use $|V(B(k))|$ processors to work simultaneously and all
          processors do the same work as follows for all $x \in V(B(k))$:
          if $x$ is a leaf node then
          Use DFS based algorithm in [24] to compute $c(x, y)$ and initialize
          $F[x][y][1] \leftarrow 0, F[x][y][0] \leftarrow w(x) \cdot c(x, y)$ for any $y \in \pi(s, x)$;
Step 2. for $k$ from 1 up to $h$ do
          Use $|V(B(k))|$ processors to work simultaneously and all
          processors do the same work as follows for all $x \in V(B(k))$:
          if $x$ is a nonleaf node then
             for each $q$ from 0 up to $p$ do
                if $0 \le q \le \min\{p, |V(B_x)|\}$ then
                   Use DFS based algorithm in [24] to compute $c(x, y)$ and
                   then compute $F[x][y][q]$ by (6) for any $y \in \pi(s, x)$;
```

Algorithm 1: SUB.

a three-children star has height of 2, the height of T^{Δ} is $2K = 2 \cdot ((n-1)/3) = 2(n-1)/3$.

Therefore, we obtain $|V_b| = n + ((5/3)n - 11/3) = (8/3)n - (11/3)$ in the worst case and conclude that BINY spends $O(n)$ time to transform T into $b(T)$. □

Theorem 5. *CLP in a general tree T is equivalent to CLP in $b(T)$.*

Proof. This theorem is equivalent to the proposition that no cache is located at one dummy node in any optimal solution to CLP in $b(T)$. Suppose that a cache is located at a dummy node α in an optimal solution L and α is added by transforming $star(x)$ by BINY. The cost of a new solution L' obtained by replacing α with x into L is less than the cost of L. This causes a contradiction. □

The binary tree obtained by applying Tamir's algorithm [15] to a general tree T with n nodes has a height of at most $n-2$, while one obtained by applying BINY to T has a height of at most $2(n-1)/3$. In terms of height of binary tree, BINY is superior to Tamir's algorithm. This will help to reduce the running time of algorithm SUB (Algorithm 1) shown in Section 3.3.

3.3. Algorithm for CLP on Trees. By Theorem 5, we only need to discuss CLP on binary trees. Let B be a binary tree and s the server (root) node, and let $V(\cdot)$ denote the node set of B or its subtree. For any node x of B, we let B_x denote the subtree of B rooted at x. Let x_l and x_r be the left child and right child of x, respectively, and then let B_{x_l} (resp. B_{x_r}) denote the subtree of B_x rooted at x_l (resp. x_r). We use h to denote the height of B and $1, 2, \ldots, h$ to label the levels of B bottom-up, and we use $B(k)$ to denote the kth level of B. Let $F[x][y][q]$ denote the minimum cost of the subproblem of p-CLP on B_x when the closest cache to x on $\pi(x, s)$ is located at y and q caches are placed in B_x. Similar to the idea of solving the p-proxy problem in [18], we propose our way of computing $F[x][y][q]$ and giving a new proof, shown in Theorem 6.

Theorem 6. *For each node x of B other than leaves, each node y on $\pi(x, s)$, and each $0 \le q \le \min\{p, |V(B_x)|\}$, one has*

$$F[x][y][q] = \min \begin{cases} \min\limits_{0 \le q_l \le q-1, q_r = q-q_l-1} \{F[x_l][x][q_l] \\ \qquad + F[x_r][x][q_r]\}, \\ \min\limits_{0 \le q_l \le q, q_r = q-q_l} \{F[x_l][y][q_l] + F[x_r][y][q_r] \\ \qquad + w(x) \cdot c(x, y)\}. \end{cases} \tag{6}$$

Proof. For any node x of B, we need to consider whether a cache is located at x or not when discussing the subproblem of p-CLP on B_x.

(1) If a cache is located at x, then x needs no paying for its overall demand. So, the cost of the subproblem of p-CLP on B_x is equal to the sum of that on B_{x_l} and that on B_{x_r}. When the closest cache to x on $\pi(s, x)$ is located at y, we observe that x is the closest cache on $\pi(s, x)$ to x_l and x_r. The possible number q of caches placed in B_x is at most p while at most $|V(B_x)|$. The number q_l of caches in B_{x_l} plus the number q_r of caches in B_{x_r} is equal to $q-1$. The key work is to find the optimal (q_l, q_r)-CAS with $q_l + q_r = q - 1$, from which the cost of the subproblem of p-CLP on B_x results is equal to

$$\min_{0 \le q_l \le q-1, q_r = q-q_l-1} \{F[x_l][x][q_l] + F[x_r][x][q_r]\}. \tag{7}$$

(2) If no cache is located at x, then the cost of x paying for its overall demand is $w(x) \cdot c(x, y)$. So, the cost of the subproblem of p-CLP on B_x is equal to the sum

of that on B_{x_l} and that on B_{x_r} and the cost of x. When the closest cache to x on $\pi(s,x)$ is located at y, we see that y is also the closest cache on $\pi(s,x)$ to x_l and x_r. As discussed above, $q \le \min\{p, |V(B_x)|\}$. The number q_l of caches in B_{x_l} plus the number q_r of caches in B_{x_r} is equal to q. The key work is to find the optimal (q_l, q_r)-CAS with $q_l + q_r = q$, from which the cost of the subproblem of p-CLP on B_x results is equal to

$$\min_{0\le q_l\le q,\ q_r=q-q_l} \{F[x_l][y][q_l] + F[x_r][y][q_r] + w(x)\cdot c(x,y)\}. \quad (8)$$

□

Obviously, the case of $q < 0$ is forbidden. Let $F[x][y][q] = \infty$, $\forall q < 0$ for each node x of B and each node y on $\pi(s,x)$. From the definition of $F[x][y][q]$, we know that $F[s][s][p]$ is just the minimum cost of p-CLP. Initially, we set $F[x][y][0] = w(x)\cdot c(x,y)$ and $F[x][y][1] = 0$ for each leaf node x of B and each node y on $\pi(s,x)$.

We can first use the *depth-first search* (DFS) based algorithm in [24] to traverse B, by which we can record the parent node of every node x (thus record $\pi(s,x)$ step by step) and compute the cost of path connecting any node and its ancestor. Based on Theorem 6 and above discussions, we devise a bottom-up dynamic programming algorithm, which can be described as a parallel algorithm SUB by using the techniques in [25].

Theorem 7. *Given any binary tree B with n nodes and a height of h,* SUB *runs in $O(ph^2 + n)$ time for computing p-CLP on B.*

Proof. Step 0 uses DFS based algorithm in [24] to traverse B, which runs $O(n)$ time. In Step 1, for each level $k = 1, 2, \ldots, h$, the processor at every leaf node $x \in V(B(k))$ uses the algorithm in [24] to make initialization for each $y \in \pi(s,x)$, which takes $O(h)$ time. So, Step 1 runs at most $O(h^2)$ time. In Step 2, for each level $k = 1, 2, \ldots, h$, the processor at every nonleaf node $x \in V(B(k))$ uses the algorithm in [24] to compute $c(x,y)$ and then compute $F[x][y][q]$ by (6) for every $0 \le q \le \min\{p, |V(B_x)|\}$ and every $y \in \pi(s,x)$, which takes at most $O(p^2 h)$ time. So, Step 2 runs at most $O(p^2h^2)$ time. We can use the method in [15] to infer that the practical running time of Step 2 is $O(ph^2)$. Therefore, SUB runs in $O(ph^2 + n)$ time. □

3.4. Algorithm for $(p,2)$-CSOP CLP on General Graphs. Based on (5) and discussions therein, we can solve $(p,2)$-CSOP CLP on a general graph G by first computing $C(T(s_1), p_1)$ in $T(s_1)$ and $C(T(s_2), p_2)$ in $T(s_2)$ for any (p_1, p_2)-CAS with $p_1 + p_2 = p$ and then determining an optimal (p_1, p_2)-CAS such that the sum of $C(T(s_1), p_1)$ and $C(T(s_2), p_2)$ is minimized. We discover that the output $F[s_i][s_i][p_i]$, $i = 1, 2$, of SUB is just the value of $C(T(s_i), p_i)$ when we apply SUB to $T(s_i)$. This forms our algorithm for $(p,2)$-CSOP CLP on a general graph, described as algorithm GLOB (Algorithm 2).

Suppose that $T(s_1)$ has n_1 nodes and $T(s_2)$ has n_2 nodes. Clearly, $n_1 + n_2 = n$. GLOB uses BINY to transform $T(s_1)$ into $b(T(s_1))$ and $T(s_2)$ into $b(T(s_2))$. From Theorem 4, we know that $b(T(s_i))$, $i = 1, 2$, has at most $(8/3)n_i - 11/3$ nodes including at most $(5/3)n_i - 11/3$ dummy nodes and has a height of at most $2(n_i - 1)/3$. Let h_i denote the height of $b(T(s_i))$. For any (p_1, p_2)-CAS with $p_1 + p_2 = p$, GLOB applies SUB to p_1-CLP on $b(T(s_1))$ and p_2-CLP on $b(T(s_2))$, respectively. It follows from Theorem 7 that the former takes at most $O(p_1 h_1^2 + (8/3)n_1)$ time and the latter takes at most $O(p_2 h_2^2 + (8/3)n_2)$ time. Therefore, the running time of GLOB is at most

$$\begin{aligned}&\sum_{p_1=0}^{p} O\left(\left(p_1 h_1^2 + \frac{8}{3}n_1\right) + \left(p_2 h_2^2 + \frac{8}{3}n_2\right)\right)\\ &= \sum_{p_1=0}^{p} O\left(p_1 h_1^2 + \frac{8}{3}n_1 + (p - p_1) h_2^2 + \frac{8}{3}(n - n_1)\right) \quad (9)\\ &\le O\left(p^2 \cdot \max\{h_1^2, h_2^2\} + \frac{8}{3}np\right).\end{aligned}$$

Further, we take $h_1 \le 2(n_1 - 1)/3$ and $h_2 \le 2(n_2 - 1)/3$ into the above inequality to obtain that the running time of GLOB in the worst case is at most

$$\begin{aligned}&O\left(p^2 \max\left\{\frac{4(n_1-1)^2}{9}, \frac{4(n_2-1)^2}{9}\right\} + \frac{8}{3}np\right)\\ &\le O\left(\frac{4}{9}p^2 n^2\right).\end{aligned} \quad (10)$$

Theorem 8. *Given an undirected graph $G = (V, E, w, c)$ with n nodes and two server nodes,* GLOB *runs in $O(p^2 \max\{h_1^2, h_2^2\} + (8/3)np)$ time for $(p,2)$-CSOP CLP on G and runs in at most $O((4/9)p^2n^2)$ time in the worst case.*

4. Generalization

In this section, we discuss the *p-cache location problem with m identical servers under CSOP* (abbreviated to (p,m)-CSOP CLP) on an undirected graph $G = (V, E, w, c)$. Let $S = \{s_1, \ldots, s_m\}$ be a collection of m identical servers. Given any set $\Omega \subset V \setminus S$, we let $c(z, \Omega \cup S)$ denote the cost of node z paying for its per demand and let $f(\Omega)$ denote the total cost of all the nodes paying for their overall demand; that is, $f(\Omega) = \sum_{z\in V} w(z)c(z, \Omega \cup S)$. The aim of (p,m)-CSOP CLP is to find p cache locations in G to minimize the total cost of all the nodes paying for their overall demand. In essence, (p,m)-CSOP CLP aims to find an optimal set Ω^* to minimize the value of $f(\Omega)$; that is, $f(\Omega^*) = \min_{\Omega\subset V\setminus S} f(\Omega)$.

In (p,m)-CSOP CLP, every client connects to the closest server to itself via a shortest route and gets information from the closest cache on the route or server. Let $V(s_i)$, $1 \le i \le m$ be the subset of nodes of G to which s_i is the closest server. Let $T(s_i)$, $1 \le i \le m$ denote the single-source shortest paths tree in G with s_i as origin spanning $V(s_i)$ and let $\Omega(s_i)$ denote the

```
Input: $T(s_1)$ and $T(s_2)$ derived from $G = (V, E, w, c)$ with $s_1$ and $s_2$;
Output: $\Gamma^*$;
Step 0. Use BINY to transform $T(s_1)$ into $b(T(s_1))$ and $T(s_2)$ into $b(T(s_2))$;
Step 1. $\Delta \leftarrow \infty$;
        for $p_1$ from 0 up to $p$ do (then $p_2 = p - p_1$)
            Apply SUB to $b(T(s_i)), i = 1, 2$ to obtain $F[s_i][s_i][p_i]$;
            $C(b(T(s_i)), p_i) \leftarrow F[s_i][s_i][p_i]$;
            if $C(b(T(s_1)), p_1) + C(b(T(s_2)), p_2) < \Delta$ then
                $\Delta \leftarrow C(b(T(s_1)), p_1) + C(b(T(s_2)), p_2)$;
                $\Gamma^*(s_1) \leftarrow \Gamma(s_1), \Gamma^*(s_2) \leftarrow \Gamma(s_2)$;
Step 2. $\Gamma^* \leftarrow \Gamma^*(s_1) \cup \Gamma^*(s_2)$;
```

Algorithm 2: GLOB.

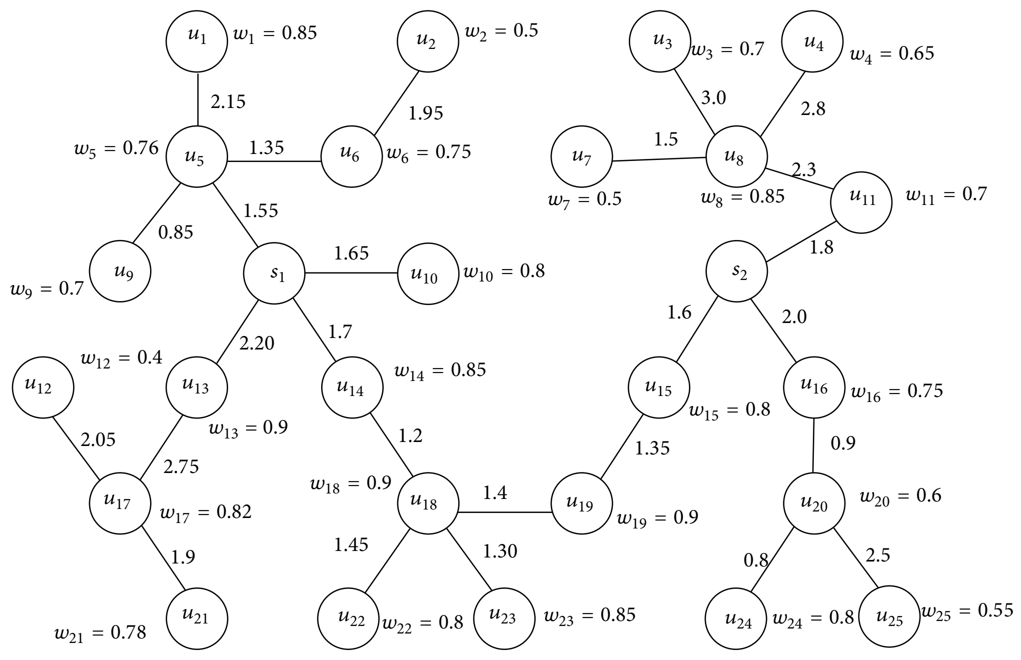


Figure 1: An example tree network.

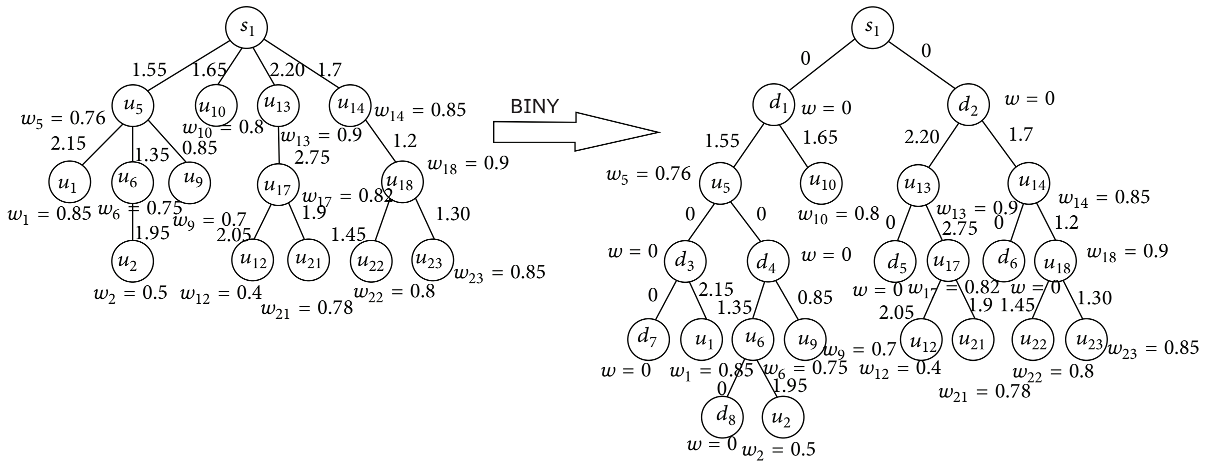


Figure 2: Illustration for $T(s_1)$ and $b(T(s_1))$.

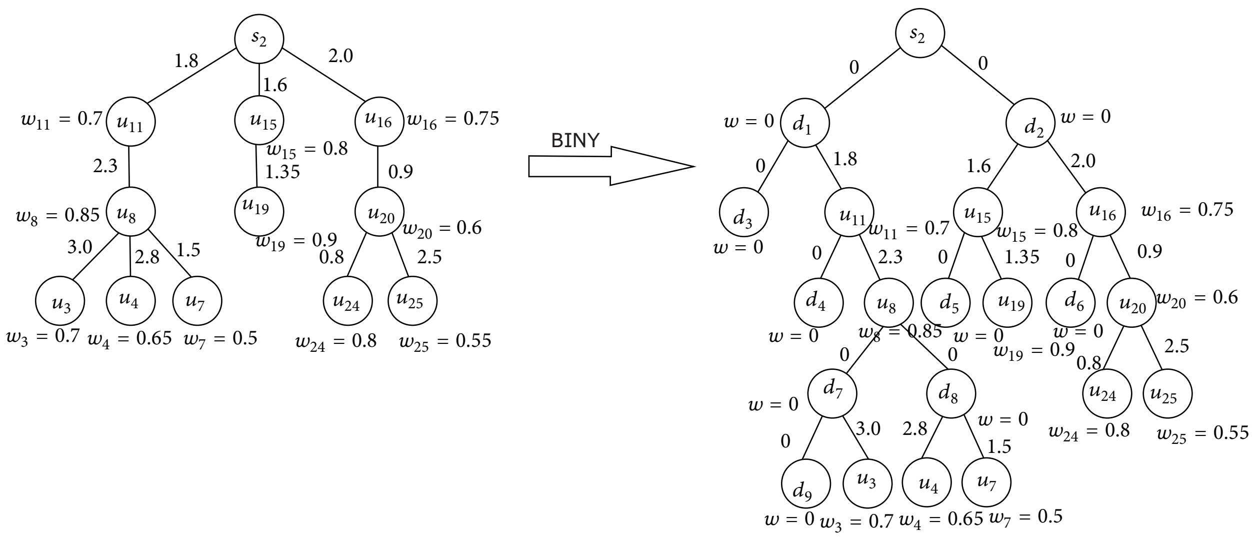
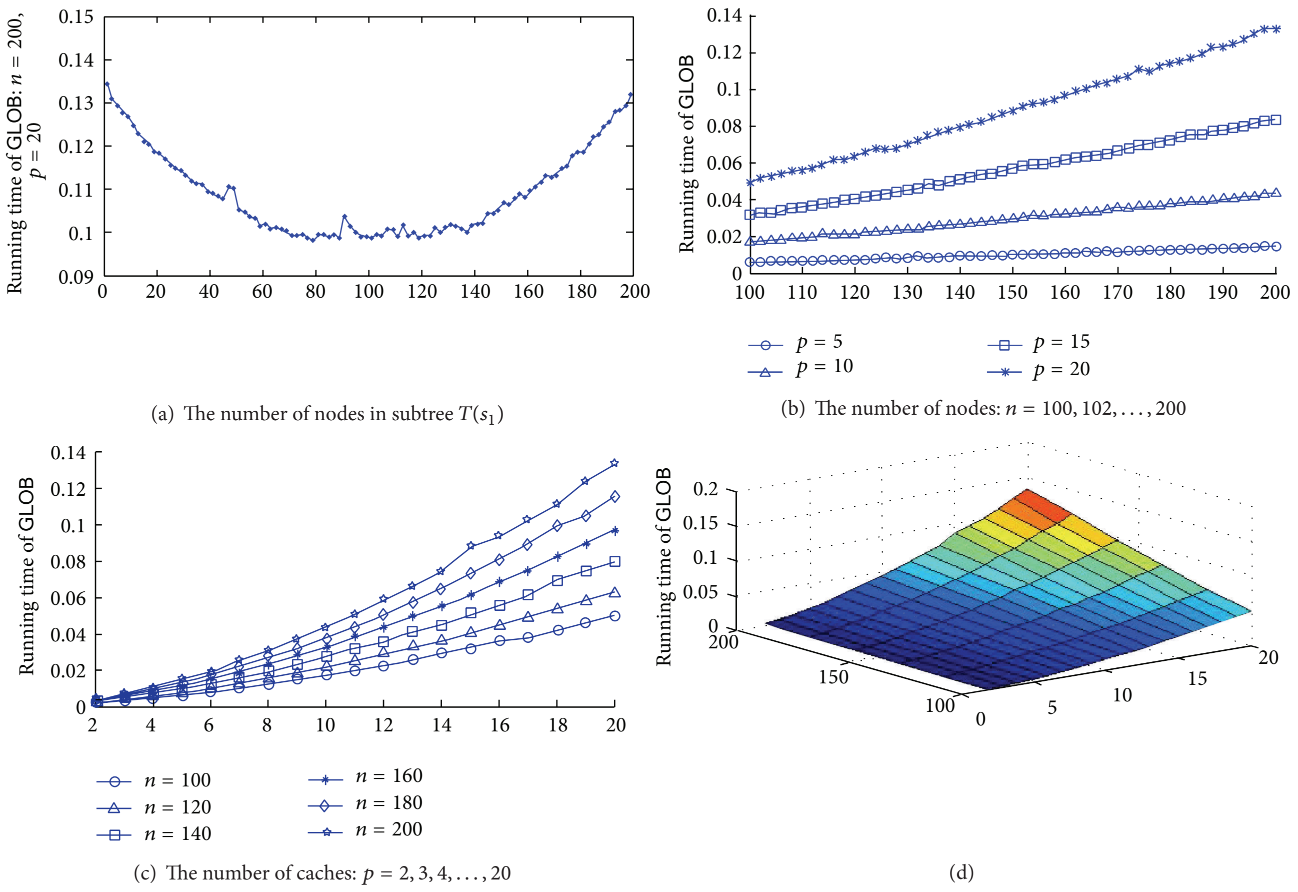


Figure 3: Illustration for $T(s_2)$ and $b(T(s_2))$.



(a) The number of nodes in subtree $T(s_1)$

(b) The number of nodes: $n = 100, 102, \ldots, 200$

(c) The number of caches: $p = 2, 3, 4, \ldots, 20$

(d)

Figure 4: Compare the running times of GLOB for $n = 100, 102, 104, \ldots, 200$ and $2 \leq p \leq 20$.

subset of caches placed in $T(s_i)$. Hence, we can rewrite $f(\Omega)$ to be

$$f(\Omega) = f\left(\bigcup_{i=1}^{m} \Omega(s_i)\right) = \sum_{i=1}^{m} \sum_{z \in T(s_i)} w(z) \cdot c\left(z, \Omega(s_i) \cup \{s_i\}\right). \tag{11}$$

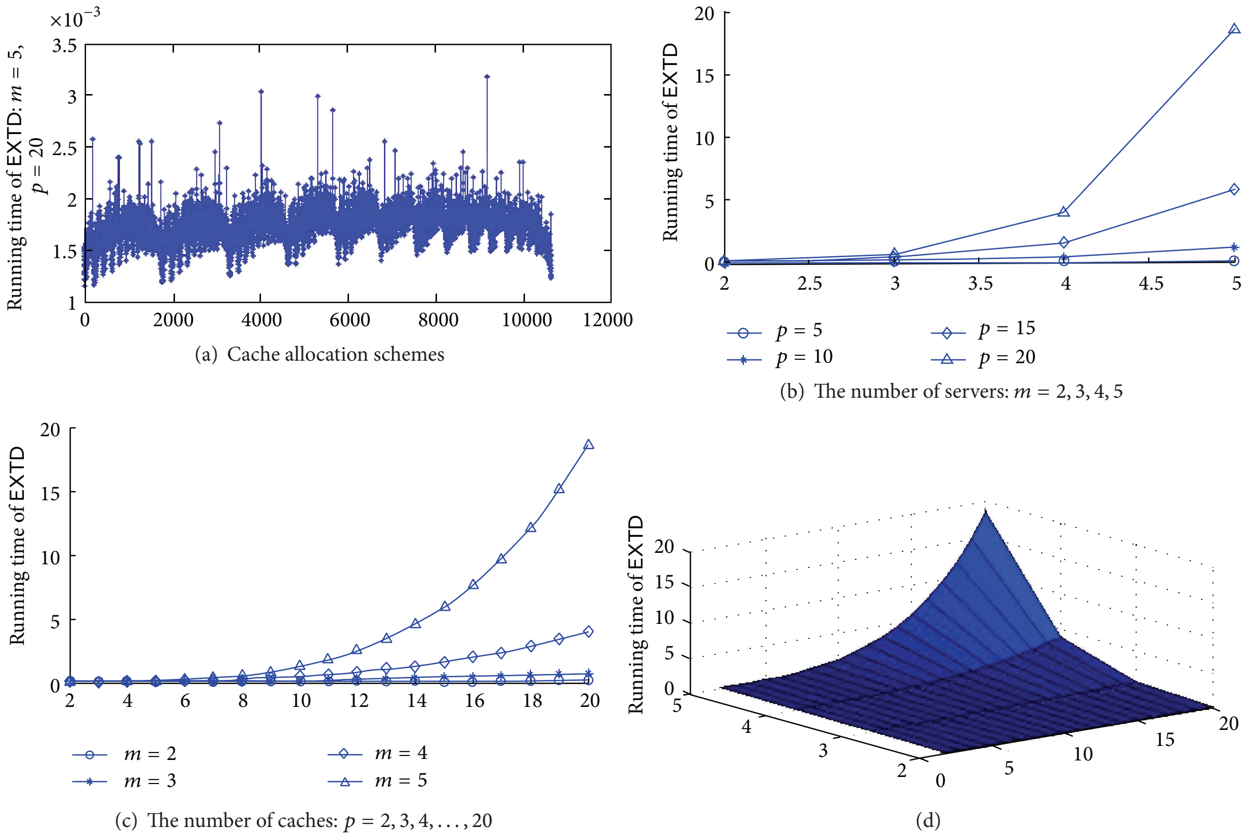


FIGURE 5: Compare the running times of EXTD for $m = 2, 3, 4, 5$ and $2 \le p \le 20$ and fixed $n = 200$ or 201.

Let p_i be the number of caches in $\Omega(s_i)$. Let $\vec{p} = (p_1, p_2, \ldots, p_m)$ with $\sum_{i=1}^{m} p_i = p$. Such a combination of $\Omega(s_i), 1 \le i \le m$ having p_i caches is denoted as $\vec{p}$-CAS. Thus, (p,m)-CSOP CLP consists of m CLPs when m servers are placed at m predesignated locations of G. For any $\vec{p}$-CAS, (p,m)-CSOP CLP consists of m subproblems; that is, p_i-CLP in $T(s_i)$, $1 \le i \le m$. Let $C(G,p)$ denote the minimum cost of (p,m)-CSOP CLP on G and let $C(T(s_i), p_i)$, $1 \le i \le m$ denote the minimum cost of p_i-CLP in $T(s_i)$. Therefore,

$$C(G,p) = \min_{\forall(p_1,\ldots,p_m)} \sum_{i=1}^{m} C(T(s_i), p_i). \tag{12}$$

Based on (12), for any $\vec{p}$-CAS with $\sum_{i=1}^{m} p_i = p$, we can solve (p,m)-CSOP CLP on G by first computing $C(T(s_i), p_i)$ in $T(s_i)$ for every $1 \le i \le m$ and then determining an optimal $\vec{p}$-CAS such that the sum of $C(T(s_i), p_i)$, $1 \le i \le m$, is minimized. This idea can be described as algorithm EXTD.

Lemma 9. *The number of $\vec{p}$-CASs in (p,m)-CSOP CLP is* $\binom{m+p-1}{p}$.

Proof. The problem of allocating p caches to m distinct subtrees can be reduced to the model of putting p same balls into m distinct boxes. We first draw $p+m-1$ dots one by one in a line and then select p dots to place balls and the other $m-1$ dots to place baffles. The line is partitioned into m sections (boxes) by these $m-1$ baffles together with two immaterial baffles at two ends of the line. There are $\binom{m+p-1}{m-1}$ ways in all to partition the line into m boxes. Every way of partitioning the line produces a $\vec{p}$-CAS. Therefore, (p,m)-CSOP CLP contains $\binom{m+p-1}{p}$ $\vec{p}$-CASs. □

For any $\vec{p}$-CAS with $\sum_{i=1}^{m} p_i = p$, the combination of Theorems 7 and 4 yields that the running time of EXTD solving all p_i-CLP in $T(s_i)$, $1 \le i \le m$, is

$$\begin{aligned}\sum_{i=1}^{m} O\left(p_i h_i^2 + \frac{8}{3} n_i\right) &= O\left(\sum_{i=1}^{m} p_i h_i^2 + \frac{8}{3}\sum_{i=1}^{m} n_i\right) \\ &\le O\left(p \cdot \max_{1\le i\le m} h_i^2 \cdot + \frac{8}{3} n\right).\end{aligned} \tag{13}$$

By Lemma 9, we conclude that the running time of EXTD is

$$\begin{aligned}&\sum_{\forall(p_1,\ldots,p_m)} \sum_{i=1}^{m} O\left(p_i h_i^2 + \frac{8}{3} n_i\right) \\ &\quad \le O\left(\left(p \cdot \max_{1\le i\le m} h_i^2 + \frac{8}{3} n\right)\binom{m+p-1}{p}\right).\end{aligned} \tag{14}$$

From $h_i \le 2(n_i - 1)/3$, $n_i \le n + 1 - m$ for every $1 \le i \le m$, we get

$$\max_{1\le i\le m} h_i^2 \le \max_{1\le i\le m} \left(\frac{2(n_i - 1)}{3}\right)^2 \le \frac{4}{9}(n-m)^2. \tag{15}$$

Table 1: Major data output by GLOB for $(4,2)$-CSOP CLP on T.

p_1	$\Gamma_1^*(s_1)$	$C(T(s_1),p_1)$	p_2	$\Gamma_2^*(s_2)$	$C(T(s_2),p_2)$	Total cost
0		37.2100	4	u_3,u_8,u_{15},u_{20}	8.5600	45.7700
1	u_{17}	27.3100	3	u_8,u_{15},u_{20}	10.6600	37.9700
2	u_{17},u_{18}	19.9150	2	u_8,u_{20}	13.3800	33.2950
3	u_5,u_{17},u_{18}	14.3970	1	u_8	19.0350	33.4320
4	u_5,u_{13},u_{17},u_{18}	12.4170	0		30.1050	42.5220

Table 2: Major data output by GLOB for $(5,2)$-CSOP CLP on T.

p_1	$\Gamma_1^*(s_1)$	$C(T(s_1),p_1)$	p_2	$\Gamma_2^*(s_2)$	$C(T(s_2),p_2)$	total cost
0		37.2100	5	$u_3,u_4,u_8,u_{15},u_{20}$	6.7400	43.9500
1	u_{17}	27.3100	4	u_3,u_8,u_{15},u_{20}	8.5600	35.8700
2	u_{17},u_{18}	19.9150	3	u_8,u_{15},u_{20}	10.6600	30.5750
3	u_5,u_{17},u_{18}	14.3970	2	u_8,u_{20}	13.3800	27.7770
4	u_5,u_{13},u_{17},u_{18}	12.4170	1	u_8	19.0350	31.4520
5	$u_1,u_5,u_{13},u_{17},u_{18}$	10.5895	0		30.1050	40.6945

Obviously, we have

$$\binom{m+p-1}{p} = \frac{(m+p-1)!}{p!\,(m-1)!} \le \frac{(m+p)^p}{p!}. \tag{16}$$

We take (15) and (16) into (14) to obtain that the running time of EXTD in the worst case is at most $O((4/9)(n-m)^2((m+p)^p/(p-1)!))$.

Theorem 10. *Given an undirected graph $G=(V,E,w,c)$ with n nodes and m servers,* EXTD *runs in $O(m^p(p\cdot\max_{1\le i\le m}h_i^2+(8/3)n))$ time for (p,m)-CSOP CLP on G and runs in at most $O((4/9)(n-m)^2((m+p)^p/(p-1)!))$ time in the worst case.*

5. Numerical Experiments

5.1. An Illustrative Example. In this subsection, we first give an example to illustrate our algorithm GLOB for computing $(p,2)$-CSOP CLP. Considering that $(p,2)$-CSOP CLP on a general network can be reduced to that on a corresponding tree network, we select a tree network as our example for ease of illustration, shown in Figure 1. The example tree T has 25 client nodes labelled by $u_1,u_2,\dots,u_{25}$ and two server nodes labelled by s_1 and s_2. The number w_k, $1\le k\le 25$, on client node u_k represents the demand account of u_k, and the number on every edge represents the cost of one node paying for its per demand. For instance, the demand account of u_5 is 0.76, and the total cost of u_5 paying for its overall demand is $1.55\times 0.76 = 1.1780$.

First, we make preparation works. The unique path $\Pi[s_1,s_2]$ of T connecting s_1 and s_2 is $s_1u_{14}u_{18}u_{19}u_{15}s_2$. It is easy to see that u_{14} and u_{18} are closer to s_1 than s_2 while u_{19} and u_{15} are closer to s_2 than s_1. Thus, $N(s_1)=\{s_1,u_{14},u_{18}\}$ and $N(s_1)=\{u_{19},u_{15},s_2\}$. Based on Lemma 2, we use the DFS based approach to obtain $T(s_1)$ and $T(s_2)$, shown in the left subfigure of Figure 2 and the left subfigure of Figure 3, respectively. Both the heights of $T(s_1)$ and $T(s_2)$ are three. We apply BINY to transform $T(s_1)$ into a binary tree $b(T(s_1))$ shown in the right subfigure of Figure 2, where eight dummy nodes added by BINY are labelled by $d_1,d_2,\dots,d_8$. Similarly, we obtain $b(T(s_2))$ shown in the right subfigure of Figure 3, where nine dummy nodes are labelled by $d_1,d_2,\dots,d_9$. All the dummy nodes have a weight of zero and all the edges between a dummy node and its parent node have a weight of zero. Both the height of $T(s_1)$ and $T(s_2)$ are five.

Next, we use GLOB to solve $(p,2)$-CSOP CLP on T. Set $p=4$ and $p=5$ as examples. For any (p_1,p_2)-CAS with $p_1+p_2=4$, GLOB computes the minimum cost $C(T(s_1),p_1)$ of p_1-CLP in $T(s_1)$ and the set $\Gamma_1^*(s_1)$ of cache locations and the minimum cost $C(T(s_2),p_2)$ of p_2-CLP in $T(s_2)$ and $\Gamma_1^*(s_2)$. The data are listed in Table 1. Clearly, the optimal value $C(T,4)$ of $(4,2)$-CSOP CLP on T is 33.2950, and thus the optimal set Γ_1^* of cache locations is $\{u_8,u_{17},u_{18},u_{20}\}$. Similarly, the data output by GLOB for any (p_1,p_2)-CAS with $p_1+p_2=5$ are listed in Table 2. Clearly, the optimal value $C(T,5)$ of $(5,2)$-CSOP CLP on T is 27.7770, and thus the optimal set Γ_2^* is $\{u_5,u_8,u_{17},u_{18},u_{20}\}$.

5.2. Comparison of Running Times. In this subsection, we make a large number of numerical experiments to compare the running times of our algorithm GLOB and EXTD, respectively. In view of the fact that (p,m)-CSOP CLP with $m\ge 2$ on a general graph can be reduced to multiple CLPs on binary trees, we select a series of *complete binary trees* as examples for ease of comparison. All the binary trees are generated randomly and have almost same number of nodes. We build a centralized parallel computer system (i.e., a star network with one central computer and five parallel computers) by connecting six identical PCs equipped with 2 GB RAM and Intel core i5 CPU using a Windows 7 operating system. Our numerical experiments were carried out on this computer system.

For $(p,2)$-CSOP CLP, we consider different inputs of n and p: $n = 100,102,104,\dots,200$ and $2\le p\le 20$. All the binary trees we select have odd nodes. Given $n=200$ and $p=20$, there are 100 different combinations of n_1 and

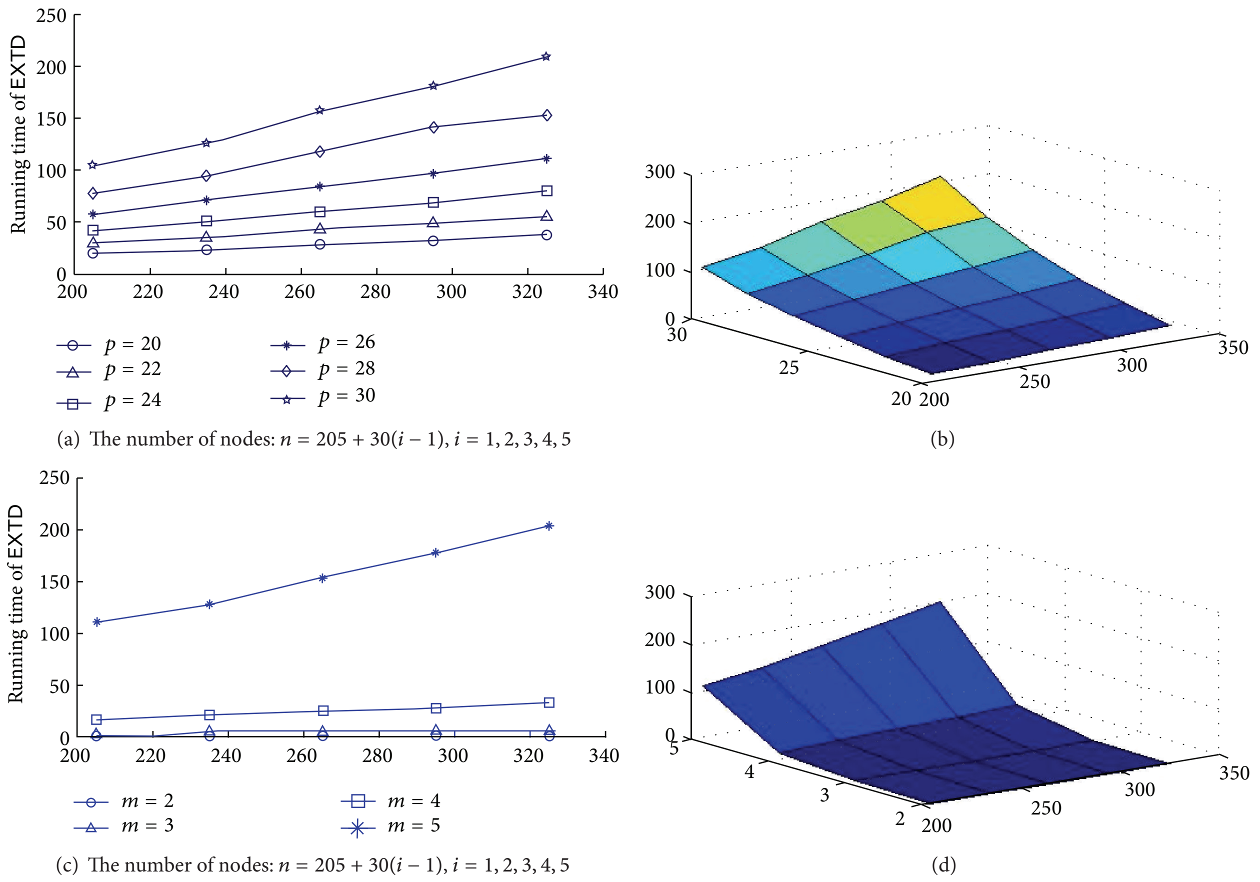


FIGURE 6: Compare the running times of EXTD for $n = 205 + 30 \times (i - 1)$, $1 \le i \le 5$ and $m = 2, 3, 4, 5$ and $p = 20, 22, 24, 26, 28, 30$.

n_2; that is, $n_1 = 1, 3, 5, \ldots, 199$ and $n_2 = 200 - n_1$. All the running times of GLOB for 100 combinations are depicted in Figure 4(a). Given any combination of n and p, we record the average running time of 100 combinations. In Figure 4(b), all the average times for the combinations of n and fixed p are depicted, for any given $p = 5, 10, 15, 20$. In Figure 4(c), all the average times for the combinations of p and fixed n are depicted, for any given $n = 100, 120, 140, 160, 180, 200$. In Figure 4(d), all the average times for the combinations of n and p are depicted.

For (p, m)-CSOP CLP, we are given $n = 200$ or 201 and consider different inputs of m and p: $2 \le m \le 5$ and $2 \le p \le 20$. All the binary trees we select have odd nodes. Given $m = 5$ and $p = 20$, there are 10626 different CAS's. All the running times of EXTD for 10626 CAS's are depicted in Figure 5(a). Given any combination of m and p, we record the average running time of 10626 CAS's. In Figure 5(b), all the average times for the combinations of m and fixed p are depicted, for any given $p = 5, 10, 15, 20$. In Figure 4(c), all the average times for the combinations of p and fixed m are depicted, for any $m = 2, 3, 4, 5$. In Figure 5(d), all the average times for the combinations of m and p are depicted.

Next, we present the running times of EXTD for different inputs of n. Given $m = 5$, we consider six different combinations of $p = 20, 22, 24, 26, 28, 30$ and fixed $m = 5$. In Figure 6(a), all the average times for $n = 205 + 30 \times (i - 1)$, $1 \le i \le 5$, are depicted. Also, we depict all the average times for the combinations of n and p in Figure 6(b). Note that every complete binary tree has $41 + 6 \times (i - 1)$ nodes. Given $p = 30$, we consider four different combinations of $m = 2, 3, 4, 5$ and fixed $p = 30$. In Figure 6(c), all the average times for exactly or approximately $n = 205 + 30 \times (i - 1)$, $1 \le i \le 5$ are depicted. In Figure 6(d), all the average times for the combinations of n and m are depicted.

6. Conclusions

In this paper, we presented an efficient algorithm GLOB for $(p, 2)$-CSOP CLP and EXTD for (p, m)-CSOP CLP based on a fast parallel algorithm for CLP, respectively. GLOB runs in a polynomial time while EXTD applies to the cases of not too large p and m in general. Fortunately, both m and p are not so large in practice that the running time of EXTD becomes intolerable with n increasing fast. Therefore, we think that EXTD will have extensive applications.

Every cache involved in this paper is identical to server; that is, every cache has the same contents as server. However, a cache sometimes only contains one part of contents. Suppose that every cache contains ρ% contents of server. Under CSOP, each node connects to the closest server to itself via a shortest route. On average, each node has ρ% demand provided by the closest cache on the route and $(1 - \rho)$%

demand provided by the server. Hence, the cost of node z paying for its overall demand is equal to $\rho\%\cdot c(z,\Omega(s_i)\cup\{s_i\})+(1-\rho\%)\cdot c(z,s_i)$. Let $g(\Omega)$ denote the total cost of all the nodes paying for their overall demand. We have

$$\begin{aligned} g(\Omega) &= g\left(\bigcup_{i=1}^{m}\Omega(s_i)\right) \\ &= \sum_{i=1}^{m}\sum_{z\in T(s_i)} w(z)\left(\rho\%\cdot c\left(z,\Omega(s_i)\cup\{s_i\}\right)\right. \\ &\qquad \left.+\left(1-\rho\%\right)\cdot c\left(z,s_i\right)\right). \end{aligned} \tag{17}$$

This version of (p,m)-CSOP CLP aims to find an optimal set Ω^* such that $g(\Omega)$ is minimized. The problem remains as one future research topic.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "World wide web caching: the application-level view of the internet," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 170–178, 1997.

[2] P. Danzig, R. Hall, and M. Schwartz, "A case for caching file objects inside internetworks," *ACM SIGCOMM Computer Communication Review*, vol. 23, no. 4, pp. 239–248, 1993.

[3] S. Glassman, "A caching relay for the World Wide Web," *Computer Networks and ISDN Systems*, vol. 27, no. 2, pp. 165–173, 1994.

[4] N. Yeager and R. McGrath, *Web Server Technology: The Advanced Guide for World Wide Web Information Providers*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 1996.

[5] A. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz, and K. Worrell, "A hierarchical internet object cache," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC '96)*, p. 13, USENIX Association, Berkeley, Calif, USA, 1996.

[6] E. Cohen, B. Krishnamurthy, and J. Rexford, "Improving end-to-end performance of the Web using server volumes and proxy filters," *Computer Communication Review*, vol. 28, no. 4, pp. 241–253, 1998.

[7] M. Abrams, C. Stanridge, G. Abdulla, E. Fox, and S. Williams, "Removal policies in network caches for World-Wide Web documents," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 4, pp. 293–305, 1996.

[8] P. Krishnan, "Utility of co-operating Web proxy caches," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 195–203, 1998.

[9] R. Malpani, J. Lorch, and D. Berger, "Making world wide web caching servers cooperate," in *Proceedings of the 4th International World-Wide Web Conference*, pp. 107–117, Boston, Mass, USA, December 1995.

[10] A. Heddaya and S. Mirdad, "WebWave: globally load balanced fully distributed caching of hot published documents," in *Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97)*, pp. 160–168, Baltimore, Md, USA, May 1997.

[11] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[12] B. Li, X. Deng, M. Golin, and K. Soharby, "On the optimal placement of Web proxies in the Internet: linear topology," in *Proceedings of the IFIP TC-6 8th International Conference on High Performance Networking (HPN '98)*, pp. 485–495, Vienna, Austria, September 1998.

[13] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the Internet," in *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societie (INFOCOM '99)*, pp. 1282–1290, New York, NY, USA, March 1999.

[14] O. Kariv and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems. II: the p-Medians," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 539–560, 1979.

[15] A. Tamir, "An O(pn^2) algorithm for the p-median and related problems on tree graphs," *Operations Research Letters*, vol. 19, no. 2, pp. 59–64, 1996.

[16] G. J. Woeginger, "Monge strikes again: optimal placement of web proxies in the internet," *Operations Research Letters*, vol. 27, no. 3, pp. 93–96, 2000.

[17] G. Chen, G. Zhang, and W. Ding, "Web proxy location problem in the tree networks," *Applied Mathematics A: Journal of Chinese Universities*, vol. 19, pp. 510–514, 2004 (Chinese).

[18] G. Chen, G. Zhang, and R. E. Burkard, "The web proxy location problem in general tree of rings networks," *Journal of Combinatorial Optimization*, vol. 12, no. 4, pp. 327–336, 2006.

[19] D. Du, "Placement of read-write Web proxies in the Internet," in *Proceedings of the The 21st International Conference on Distributed Computing Systems (ICDCS '01)*, pp. 687–687, Phoenix, Ariz, USA, April 2001.

[20] X. Jia, D. Li, X. Hu, W. Wu, and D. Du, "Placement of Web-server proxies with consideration of read and update operations on the Internet," *Computer Journal*, vol. 46, no. 4, pp. 378–390, 2003.

[21] J. Liu and J. Yang, "Delay-constrained Web proxy problem in Internet," *Computer Engineering and Applications*, vol. 45, no. 31, pp. 109–110, 2009.

[22] J. A. Bondy and U. S. R. Murty, *Graph Theory with Application*, Macmillan, London, UK, 1976.

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[24] W. Ding, Y. Zhou, G. Chen, H. Wang, and G. Wang, "On the 2-MRS problem in a tree with unreliable edges," *Journal of Applied Mathematics*, vol. 2013, Article ID 743908, 11 pages, 2013.

[25] W. Ding and G. Xue, "A fast parallel algorithm for finding a most reliable source on a general ring-tree graph with unreliable edges," in *Combinatorial Optimization and Applications*, vol. 6831 of *Lecture Notes in Computer Science*, pp. 98–112, Springer, Heidelberg, Germany, 2011.