



A Parallel Hill Climbing Algorithm for Pushing Dependent Data in Clients–Providers–Servers Systems

FRANCISCO JAVIER OVALLE-MARTÍNEZ
SITE, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada

JULIO SOLANO GONZÁLEZ*
DISCA, IIMAS, UNAM, P.O. Box 20-726, Del. A. Obregon, Mexico D.F. 01000, Mexico

IVAN STOJMENOVIĆ
SITE, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada

Abstract. The up-link bandwidth in satellite networks and in advanced traffic wireless information system is very limited. A server broadcasts data files provided by different independent providers and accessed by many clients in a round-robin manner. The clients who access these files may have different patterns of access. Some clients may wish to access several files in any order (AND), some wish to access one out of several files (OR), and some clients may access a second file only after accessing another file (IMPLY). The goal of the server is to order the files in a way that minimizes the access time of the clients given some a priori knowledge of their access patterns. An appropriate clients–servers model was recently proposed by Bay-Noy, Naor and Schieber. They formulated three separate problems and proposed an algorithm that evaluates certain number of random permutations and chooses the one whose access time is minimized. In this paper, we formulate a combined AOI (AND-OR-IMPLY) problem, and propose to apply a parallel hill climbing algorithm (to each of the four problems), which begins from certain number of random permutations, and then applies hill climbing technique on each of them until there is no more improvement. The evaluation time of neighboring permutations generated in hill climbing process is optimized, so that it requires $O(n)$ time per permutation instead of $O(n^2)$ time required for evaluating access time of a random permutation, where n is the number of files the server broadcasts. Experiments indicate that the parallel hill climbing algorithm is $O(n)$ times faster than random permutations method, both in terms of time needed to evaluate the same number of permutations, and time needed to provide a high quality solution. Thus the improvement is significant for broadcasting large number of files.

Keywords: data broadcasting, hill climbing algorithms

1. Introduction

The communication channel between a server and a client is in many environments asymmetric, meaning that there is significantly more bandwidth from the server to the client than from client to the server. Thus, the data is disseminated (pushed) from a server to many clients instead of being brought (pulled) from server to client directly upon request. Examples include a satellite network in which satellites broadcast popular web pages, to speed up web surfing, a wireless network with significantly higher data transmission rate of base station compared to mobile users, advanced traffic information systems, in which hundreds or thousands of motorists may simultaneously require data from a server in a timely manner, and cable television or video on demand service with high downstream bandwidth and very limited upstream (client to server) capacity.

Push systems are proposed for asymmetric communication. The key idea is that the servers continuously and repeatedly broadcast data simultaneously to multiple clients, based on their knowledge of clients' preferences. In effect, the broadcast channel becomes a "disk" from which clients

can retrieve data as it goes by. Clients listen to the broadcast channel until they access the data they are looking for. The goal of the servers is to compute a schedule that minimizes the average access time of the clients, thereby attracting clients for future accesses. The problem of efficiently scheduling the files is also known as the broadcast disks problem. It was studied recently in literature [1,3–5,7–12,14,16]. A survey of data broadcast algorithms with various problem statements is given in [17]. It reviews access efficiency and power conservation protocols for push-based, on-demand, and hybrid data scheduling, and air indexing.

Bar-Noy, Naor and Schieber [5] studied clients–providers–servers model in which clients serve other providers who need their broadcasting capabilities. The servers push the data files to the clients, but they do not choose the files to be broadcast. They merely "sell" broadcast slots to providers who wish to spread their data. Providers select the most popular and successful servers and clients select servers that broadcast the data they are looking for while minimizing their access time. The goal of servers is therefore to minimize or improve the average access time of the clients based on some a priori knowledge, without changing the frequency of transmitting specific files (that is, violating contracts with providers).

* Corresponding author.

In the mentioned examples, satellite channel may sell broadcast slots to web pages providers, and a cable television company may sell time slots to movie producers. In the traditional model where servers are the providers, they could decide how frequently to broadcast each of the files. This is not the case with the clients–providers–servers model. Here, the only flexibility left for the servers is to fix the order of broadcasting various data files. Thus a schedule in the model introduced in [5] is a *permutation* of data files. Obviously, if server has no a priori knowledge about the most frequent files requested by clients, it can do no better than choosing an arbitrary permutation to broadcast the files. Moreover, this is true even if the server knows the popularity of each individual file, since it cannot broadcast the more popular files more often.

The authors [5] showed that knowing a priori distribution of the clients access to *pairs* of files could improve the performance of the server. Three variants are considered in [5]. In the first AND variant, clients need to access one file or two files in any order. In the second OR variant, clients wish to access either one of the two files, since both files contain needed information. In the third IMPLY variant, clients need to access two files in a certain order. Such an a priori knowledge can improve the average access time, by choosing an appropriate permutation of data files for broadcast order (examples are given in [5]). The authors [5] presented a lower bound on the average access time, and discussed the performance of two proposed algorithm. One algorithm is to select a certain number of random permutations, evaluate access time for each of them, and choose the best one. The other is a polynomial time deterministic algorithm to select a certain number of permutations that “simulates” the set of all permutations. The second algorithm guarantees the same bounds as the randomized one in the worst-case. However, as they comment themselves, their bounds and performance are tight in case of equal probabilities (in which case any permutation will have the same access time), while better lower and upper bounds for an optimal solution are needed for a given set of probabilities. The authors did not evaluate the performance of their algorithms experimentally.

In this paper, we describe a combined AOI (AND-OR-IMPLY) problem, and thus consider four different problems. In order to investigate and improve performance further, and evaluate the performance experimentally, we implemented the algorithm from [5] that evaluated certain number of random permutations (the deterministic algorithm was not implemented since it does not improve the performance [5] over random permutations algorithm), and proposed a new algorithm. The proposed algorithm is a parallel hill climbing algorithm. In this algorithm, certain number of initial random permutations is generated. Starting from each initial permutation, hill climbing algorithm is applied, which generates certain number of neighboring permutations (using a two element swap to define neighbors) for currently best permutation, and repeats this process until no improvement is made in certain number of attempts. In addition, we optimized the time needed to evaluate access time of a neighboring permuta-

tion, using already evaluated access time of current permutation. The experimental results show significant improvement in the speed and access time for the parallel hill climbing algorithm over random permutations one. More precisely, it produces high quality access time $O(n)$ times faster than random permutations method [5], where n is the number of files to be broadcast.

2. Problem statements

Let $F = \{F_0, F_1, \dots, F_{n-1}\}$ be a collection of equal size files. Assume that it takes one unit time (a slot) to broadcast each file. Define the following three sets of probabilities for three types of problems we consider [5]:

- (A) In the AND problem for $0 \leq i \leq j \leq n - 1$, the probability of accessing F_i and F_j is a_{ij} . The probability of accessing only F_i is a_{ii} . There are $n(n + 1)/2$ different probabilities, and $\sum_{j=0}^{n-1} \sum_{i=0}^j a_{ij} = 1$.
- (B) In the OR problem, for $0 \leq i \leq j \leq n - 1$, the probability of accessing F_i or F_j is b_{ij} . The probability of accessing only F_i is b_{ii} . There are $n(n + 1)/2$ different probabilities, and $\sum_{0 \leq i \leq j \leq n-1} b_{ij} = 1$.
- (C) In the IMPLY problem, for $0 \leq i \neq j \leq n - 1$, the probability of accessing F_i and then F_j is c_{ij} . The probability of accessing only F_i is c_{ii} . There are n^2 different probabilities, and $\sum_{j=0}^{n-1} \sum_{i=0}^{n-1} c_{ij} = 1$.

An infinite cyclic schedule is defined by a permutation $\pi = (\pi(0), \pi(1), \dots, \pi(n - 1))$ on the numbers $(0, 1, \dots, n - 1)$. At time T the server broadcasts file $F_{\pi(i)}$ for one unit of time where $i = T \bmod n$. Let $\sigma = \pi^{-1}$, i.e., $\sigma(\pi(i)) = i$ for all $0 \leq i \leq n - 1$. We say that files F_i and F_j are at distance d_{ij} if $d_{ij} = \min\{((\sigma(i) - \sigma(j)) \bmod n), ((\sigma(j) - \sigma(i)) \bmod n)\}$ for $0 \leq i < j \leq n - 1$. In other words, d_{ij} is the shorter cyclic distance between i and j in the permutation π and therefore $d_{ij} \leq \lfloor n/2 \rfloor$.

We say that files F_i and F_j are at directed distance dd_{ij} if $dd_{ij} = ((\sigma(j) - \sigma(i)) \bmod n)$ for $0 \leq i, j \leq n - 1$. In other words, dd_{ij} is the number of slots between a broadcast of F_i and the next broadcast of F_j . It follows that $0 \leq dd_{ij} \leq n - 1$. Note that the directed distance is a non-symmetric function, and that $dd_{ij} + dd_{ji} = n$ for $0 \leq i \neq j \leq n - 1$.

We assume that clients arrive at random time slots during the broadcasting of the infinite schedule. In order to access a file, client must arrive before the starting time of the broadcast of this file (if client arrives in the middle of broadcasting a file, it must wait until this file is broadcast entirely).

For a permutation π , let $A_{ij}(\pi)$ be the expected waiting time for a client who wishes to access both files F_i and F_j , let $B_{ij}(\pi)$ be the expected waiting time for a client who wishes to access either file F_i or file F_j , and let $C_{ij}(\pi)$ be the expected waiting time for a client who wishes to access first file F_i and

then file F_j . Let

$$\begin{aligned} A(\pi) &= \sum_{0 \leq i \leq j \leq n-1} [(a_{ij})(A_{ij}(\pi))], \\ B(\pi) &= \sum_{0 \leq i \leq j \leq n-1} [(b_{ij})(B_{ij}(\pi))], \\ C(\pi) &= \sum_{0 \leq i \neq j \leq n-1} [(c_{ij})(C_{ij}(\pi))]. \end{aligned}$$

That is, $A(\pi)$ is the expected time for all clients who wish to access one or two files, $B(\pi)$ is the expected time for all clients who wish to access one file or one out of two files, and $C(\pi)$ is the expected waiting time for all clients who wish to access one file or two files in a certain order. In the AND problem we are looking for a permutation π that minimizes $A(\pi)$, in the OR problem we are looking for a permutation π that minimizes $B(\pi)$, and in the IMPLY problem we are looking for a permutation π that minimizes $C(\pi)$. Note that, in this problem formulation, all clients belong to one of three user groups, this three separate problems are being solved here.

Bar-Noy, Naor and Schieber [5] have derived formulas for computing $A_{ij}(\pi)$, $B_{ij}(\pi)$, $C_{ij}(\pi)$, $A(\pi)$, $B(\pi)$, and $C(\pi)$. We will only give formulas here, details can be found in [5]:

$$\begin{aligned} A_{ii}(\pi) &= B_{ii}(\pi) = C_{ii}(\pi) = \frac{n}{2} + 1, \\ C_{ij}(\pi) &= \frac{n}{2} + 1 + dd_{ij}, \\ A_{ij}(\pi) &= \left(\frac{n}{2} + 1\right) + \left(d_{ij} - \frac{d_{ij}^2}{n}\right), \\ C(\pi) &= \frac{n}{2} + 1 + \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} c_{ij} dd_{ij}, \\ A(\pi) &= \frac{n}{2} + 1 + \sum_{j=0}^{n-1} \sum_{i=0}^j a_{ij} \left(d_{ij} - \frac{d_{ij}^2}{n}\right), \\ B_{ij}(\pi) &= \left(\frac{n}{2} + 1\right) - \left(d_{ij} - \frac{d_{ij}^2}{n}\right), \\ B(\pi) &= \frac{n}{2} + 1 - \sum_{j=0}^{n-1} \sum_{i=0}^j b_{ij} \left(d_{ij} - \frac{d_{ij}^2}{n}\right). \end{aligned}$$

Bar-Noy, Naor and Schieber [5] considered three separate problems: AND, OR and IMPLY. In reality, there will be mixed population of clients, with few of them in each category, with a single ordering of files for broadcast. We propose here to combine the three problems into one, so that a total of four different problems are being considered. The combined AOI (AND-OR-IMPLY) problem will simply assign certain estimated weight to each of three problems, as follows:

$$\begin{aligned} E(\pi) &= \alpha A(\pi) + \beta B(\pi) + \gamma C(\pi), \\ \text{where } \alpha + \beta + \gamma &= 1, \alpha \geq 0, \beta \geq 0, \gamma \geq 0. \end{aligned}$$

The weights correspond to the estimated overall importance of each of three basic problems. Although it is possible to define a combined problem differently, we believe that this formulation captures the essence of a combination.

We will sometimes use X to denote one of four problems AND, OR, IMPLY, and AOI, or to denote one of A , B , or C or E . Also, the corresponding set of probabilities is then denoted by x_{ij} . The authors [5] proposed the following heuristic algorithm to find the permutation that optimizes the access time for a problem X (referring, of course, to the first three variants, but the same solution can be extended to the fourth problem). The algorithm constructs m random permutations. For each generated permutation π' , $X(\pi')$ is computed. Output is permutation π that minimizes $X(\pi)$ among generated permutations.

3. Parallel hill climbing

Hill climbing [13,15] is a simple well-known heuristic method for searching optimal solutions to problems that have huge solution spaces and do not have polynomial time exact solutions. The solution space in the problem studied in this paper consists of $n!$ permutations. In this method, the search starts from an initial solution. Certain number of neighbours of currently best solutions is generated, until either a better one is found, or the number of generated neighbours exceeds the limit. The search then continues from that better neighbour in the same fashion. The method can be formally described as follows.

Algorithm *hill-climbing*(X , *best* π , *best-access-time*).

$\pi :=$ *initial permutation* (generated at random);

best $\pi := \pi$;

best-access-time := $X(\pi)$;

$i := 0$;

repeat

$\pi :=$ *neighbour*(π); *access-time* := $X(\pi)$;

if *access-time* < *best-access-time*

then {*best* $\pi := \pi$; $i := 0$;

best-access-time := *access-time*}

else $i := i + 1$

until $i = \text{max}$

The choice of function *neighbour*(π) and parameter *max* is flexible. We have used value *max* = 30 in our experiments, while the function *neighbour*(π) selects two different indices i and j at random, and swaps the corresponding elements in permutation π . For example, let $\pi = 63872154$, and let $i = 4$, $j = 2$ are generated at random. Then the neighbouring permutation is 67832154 (the second and fourth elements are swapped).

The parallel hill climbing procedure [2,6] is a hill climbing scheme that is initiated several time, so that multiple hills are claimed in parallel. It can be described as follows.

Algorithm *parallel-hill-climbing*($X, t, total\text{-}best\pi, total\text{-}best\text{-}access\text{-}time$).

$total\text{-}best\text{-}access\text{-}time := \infty$;

for $i := 1$ **to** t **do** {

hill-climbing($X, best\pi, best\text{-}access\text{-}time$);

if $best\text{-}access\text{-}time < total\text{-}best\text{-}access\text{-}time$

then {

$total\text{-}best\pi := best\pi$;

$total\text{-}best\text{-}access\text{-}time := best\text{-}access\text{-}time$)} }

The number t of calls to hill climbing procedure is a parameter whose value varies and is given in tables with experimental results.

4. Calculating access time at neighbouring permutation

The calculation of value $X(\pi)$ according to formulas given in section 2 requires $O(n^2)$ time, since it has a double loop with indices from 1 to n (again, n is the number of files, or elements in permutation). The algorithm in [5] generates certain number of random permutations, and the evaluation of each therefore requires $O(n^2)$ time. The same calculation can be applied in our parallel hill climbing algorithm. However, the full advantage of hill climbing algorithm can be exploited to calculate $X(neighbor(\pi))$ in $O(n)$ time, using already known value of $X(\pi)$.

Suppose that the permutation $neighbor(\pi)$ is obtained from permutation π by swapping the K th and L th files in π . The values x_{ij} do not change, since they are related to file dependencies. Changes occur in distance matrices d or dd . The only values in these matrices that are affected are those with one or two indices being K or L . More precisely, d_{KL} and d_{LK} are exchanged, and d_{iK} and d_{iL} are exchanged for every other index i different from K or L . The similar exchanges are made for matrix dd . These exchanges require $O(n)$ time, and are made in order to continue hill climbing method from $neighbor(\pi)$. However, before the exchange, the value $X(\pi')$, $\pi' = neighbor(\pi)$, can be calculated from value $X(\pi)$ using the following formulas, which require $O(n)$ time each:

$$A(\pi') = A(\pi) - suma_1 + suma_2,$$

$$\begin{aligned} suma_1 &= \sum_{j=0}^{K-1} a_{Kj} \left(d_{Kj} - \frac{d_{Kj}^2}{n} \right) + \sum_{j=0}^{L-1} a_{Lj} \left(d_{Lj} - \frac{d_{Lj}^2}{n} \right) \\ &+ \sum_{i=K+1}^{n-1} a_{iK} \left(d_{iK} - \frac{d_{iK}^2}{n} \right) \\ &+ \sum_{i=L+1}^{n-1} a_{iL} \left(d_{iL} - \frac{d_{iL}^2}{n} \right) - a_{LK} \left(d_{LK} - \frac{d_{LK}^2}{n} \right), \\ suma_2 &= \sum_{j=0}^{K-1} a_{Kj} \left(d'_{Kj} - \frac{d'^2_{Kj}}{n} \right) + \sum_{j=0}^{L-1} a_{Lj} \left(d'_{Lj} - \frac{d'^2_{Lj}}{n} \right) \end{aligned}$$

$$\begin{aligned} &+ \sum_{i=K+1}^{n-1} a_{iK} \left(d'_{iK} - \frac{d'^2_{iK}}{n} \right) \\ &+ \sum_{i=L+1}^{n-1} a_{iL} \left(d'_{iL} - \frac{d'^2_{iL}}{n} \right) - a_{LK} \left(d'_{LK} - \frac{d'^2_{LK}}{n} \right), \end{aligned}$$

where $d'_{Kj} = d_{Lj}$, $d'_{Lj} = d_{Kj}$, $d'_{iK} = d_{iL}$, $d'_{iL} = d_{iK}$, $d'_{LK} = d_{KL} = d_{LK}$.

$$B(\pi') = B(\pi) + suma_1 - suma_2,$$

$$\begin{aligned} suma_1 &= \sum_{j=0}^{K-1} b_{Kj} \left(d_{Kj} - \frac{d_{Kj}^2}{n} \right) + \sum_{j=0}^{L-1} b_{Lj} \left(d_{Lj} - \frac{d_{Lj}^2}{n} \right) \\ &+ \sum_{i=K+1}^{n-1} b_{iK} \left(d_{iK} - \frac{d_{iK}^2}{n} \right) \\ &+ \sum_{i=L+1}^{n-1} b_{iL} \left(d_{iL} - \frac{d_{iL}^2}{n} \right) - b_{LK} \left(d_{LK} - \frac{d_{LK}^2}{n} \right), \\ suma_2 &= \sum_{j=0}^{K-1} b_{Kj} \left(d'_{Kj} - \frac{d'^2_{Kj}}{n} \right) + \sum_{j=0}^{L-1} b_{Lj} \left(d'_{Lj} - \frac{d'^2_{Lj}}{n} \right) \\ &+ \sum_{i=K+1}^{n-1} b_{iK} \left(d'_{iK} - \frac{d'^2_{iK}}{n} \right) \\ &+ \sum_{i=L+1}^{n-1} b_{iL} \left(d'_{iL} - \frac{d'^2_{iL}}{n} \right) - b_{LK} \left(d'_{LK} - \frac{d'^2_{LK}}{n} \right), \end{aligned}$$

$$C(\pi') = C(\pi) - suma_1 + suma_2,$$

$$\begin{aligned} suma_1 &= \sum_{j=0}^{n-1} c_{Kj} \left(dd_{Kj} - \frac{dd_{Kj}^2}{n} \right) + \sum_{j=0}^{n-1} c_{Lj} \left(dd_{Lj} - \frac{dd_{Lj}^2}{n} \right) \\ &+ \sum_{i=0}^{n-1} c_{iK} \left(dd_{iK} - \frac{dd_{iK}^2}{n} \right) \\ &+ \sum_{i=0}^{n-1} c_{iL} \left(dd_{iL} - \frac{dd_{iL}^2}{n} \right) \\ &- c_{KK} \left(dd_{KK} - \frac{dd_{KK}^2}{n} \right) - c_{LK} \left(dd_{LK} - \frac{dd_{LK}^2}{n} \right) \\ &- c_{KL} \left(dd_{KL} - \frac{dd_{KL}^2}{n} \right) - c_{LL} \left(dd_{LL} - \frac{dd_{LL}^2}{n} \right), \\ suma_2 &= \sum_{j=0}^{n-1} c_{Kj} \left(dd'_{Kj} - \frac{dd'^2_{Kj}}{n} \right) + \sum_{j=0}^{n-1} c_{Lj} \left(dd'_{Lj} - \frac{dd'^2_{Lj}}{n} \right) \\ &+ \sum_{i=0}^{n-1} c_{iK} \left(dd'_{iK} - \frac{dd'^2_{iK}}{n} \right) \\ &+ \sum_{i=0}^{n-1} c_{iL} \left(dd'_{iL} - \frac{dd'^2_{iL}}{n} \right) \end{aligned}$$

$$\begin{aligned}
 & -c_{KK} \left(dd'_{KK} - \frac{dd_{KK}^2}{n} \right) - c_{LK} \left(dd'_{LK} - \frac{dd_{LK}^2}{n} \right) \\
 & -c_{KL} \left(dd'_{KL} - \frac{dd_{KL}^2}{n} \right) - c_{LL} \left(dd'_{LL} - \frac{dd_{LL}^2}{n} \right),
 \end{aligned}$$

where $dd'_{Kj} = dd_{Lj}$, $dd'_{Lj} = dd_{Kj}$, $dd'_{iK} = dd_{iL}$, $dd'_{iL} = dd_{iK}$, $dd'_{LK} = dd_{KL} = dd_{LK}$.

Finally, $E(\pi) = \alpha A(\pi) + \beta B(\pi) + \gamma C(\pi)$.

5. Performance evaluation

The two schemes were implemented in version 5.3.1.29215a (R11.1) of Matlab. The environment is chosen because it provides some useful functions for simulations. For instance, it has built in function *randperm* to generate a random permutation.

The input probabilities are generated in the following way. Files are divided into several groups, so that probabilities for accessing two files within the same group are relatively high, while the probabilities of accessing two files from different groups are low. In order to generate such matrices, a matrix P with random values p_{ij} is first generated. The elements of matrix P , of size $n \times n$, are generated using normal distribution, that is, *randn* function from Matlab. Matrix X has elements either p_{ij} or $10^{-6}p_{ij}$. $x_{ij} = p_{ij}$ if and only if files i and j belong to the same group, otherwise $x_{ij} = 10^{-6}p_{ij}$. In case of AND and OR problems, to provide symmetric input, x_{ij} and x_{ji} are reassigned so that each receives half of their original sum (that is $x_{ij} := x_{ji} := (x_{ij} + x_{ji})/2$). Finally, all elements are normalized so that the sum of all elements in the matrix is 1. The grouping is indicated in the tables according to the following example. Let $n = 10$, and let the first group contain three elements, the second – two elements, third –

three elements and fourth – two elements. This type of permutation is indicated as [3, 5, 8, 10], where elements indicate the position of last element from each group. One can assume that, for example, each group of files corresponds to different user interests, e.g., files may contain sport, news, weather and science information, respectively. For AOI problem, in our experiments, we assumed $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$.

Table 1 presents the experimental results of comparing the two algorithms, PHC (parallel-hill-climbing) and RP (random permutations) generation [5], when the same number of permutations is evaluated in both. The parameter t in the second column is the number of calls to hill climbing procedure in PHC scheme. The size n of permutation is given in the third column, and the grouping within permutation is in the fourth column. The number of evaluated permutations in PHC is counted during the algorithm. It is given in the fifth column. Note that the count for AOI problem refers to permutations that are evaluated three times each, ones for each ingredient. We then run RP algorithm [5] by selecting and evaluating the same number of random permutations. It can be observed from table 1 that access times for both algorithms are nearly same in all cases, which is the consequence of high number of evaluated permutations and relative small values for n . It is believed that the access times are near their optimum values. The difference between the two algorithms can be observed in the processing times. The ratios of the times are in all cases between $2n$ and $3n$. That is, PHC algorithm is more than $2n$ times faster than RP algorithm [5] in evaluating the same number of permutations, hence in approaching optimal solution.

The next set of experiments was designed to find the approximate time needed to reach near optimal solution by each of algorithms, for a larger value of n . The values $n = 50$ and $t = 10$ are selected, with file grouping [7,

Table 1
Comparison of the two algorithms for the same number of evaluated permutations.

X	t	n	File grouping	# of evaluated permutations	Access time for PHC	Access time for RP	Processing time for PHC	Processing time for RP
AND	15	5	[2, 3, 5]	535	3.8002	3.8002	1.9933	21.4033
AND	30	10	[3, 5, 8, 10]	2556	6.3331	6.3370	17.7000	528.1667
AND	45	15	[2, 7, 10, 12, 15]	5343	9.2579	9.4564	46.1567	2110.9333
AND	60	20	[4, 9, 12, 15, 20]	9314	11.801	12.3500	86.0333	6304.4667
AND	75	25	[5, 8, 13, 20, 25]	14652	14.6831	15.6395	187.2400	16137
OR	15	5	[2, 3, 5]	522	2.9362	2.9362	1.8133	20.3
OR	30	10	[3, 5, 8, 10]	2448	5.0986	5.1019	12.23	373.1
OR	45	15	[2, 7, 10, 12, 15]	5293	6.6188	6.6563	38.0267	1827.5
OR	60	20	[4, 9, 12, 15, 20]	8196	8.4513	8.5464	80.6301	5561.4
OR	75	25	[5, 8, 13, 20, 25]	11651	9.7356	9.9219	146.6499	11727.3
IMPLY	15	5	[2, 3, 5]	603	4.1181	4.1181	1.8100	12.5
IMPLY	30	10	[3, 5, 8, 10]	2248	8.0140	8.1899	10.4167	178.9
IMPLY	45	15	[2, 7, 10, 12, 15]	5273	12.4061	12.6271	39.2033	1126.6
IMPLY	60	20	[4, 9, 12, 15, 20]	8661	16.6485	17.5803	69.4299	2722.5
IMPLY	75	25	[5, 8, 13, 20, 25]	11745	21.507	22.6458	118.8099	6377.1
AOI	15	5	[2, 3, 5]	553	3.43178	3.43178	5.6166	54.2033
AOI	30	10	[3, 5, 8, 10]	2417	6.05203	6.09003	40.3467	1080.1667
AOI	45	15	[2, 7, 10, 12, 15]	5303	8.56799	8.69049	123.3867	5065.0333
AOI	60	20	[4, 9, 12, 15, 20]	8724	11.09565	11.49426	236.0933	14588.3667
AOI	75	25	[5, 8, 13, 20, 25]	12683	13.57413	14.18196	452.6998	34241.4

Table 2
Processing times and access times for two algorithms after given number of evaluated permutations, for $n = 30$ and grouping [2, 7, 10, 15, 18, 20, 24, 26, 30].

X	Number of evaluated permutations	Access time for HC	Access time for RP	Processing time for HC	Processing time for RP	Ratio of processing times
AND	1	19.2584	18.7717	2.04	1.76	0.86
AND	10	19.1522	19.6425	2.15	22.79	10.60
AND	20	18.9267	19.4503	2.26	52.4	23.19
AND	30	18.7324	19.061	2.37	77.77	32.81
AND	40	18.837	19.3958	2.53	101.5	40.12
AND	50	18.6846	19.4252	2.58	125.12	48.50
AND	60	18.8278	19.7333	2.69	149.18	55.46
AND	70	18.4882	19.7308	2.8	173.67	62.03
AND	80	18.289	19.4295	2.96	197.35	66.67
AND	90	18.6088	19.2994	3.07	221.68	72.21
AND	100	17.9332	18.9751	3.24	245.79	75.86
OR	1	12.4379	12.6252	2.63	2.42	0.92
OR	10	12.2202	12.1847	2.85	34.55	12.12
OR	20	12.3375	12.3682	3.07	76.89	25.05
OR	30	12.0763	12.6406	3.24	103.2	31.85
OR	40	12.4011	12.6225	3.46	128.2	37.05
OR	50	12.3512	12.3591	3.68	151.21	41.09
OR	60	12.3178	12.2652	3.9	174.23	44.67
OR	70	12.3276	12.5527	4.06	197.46	48.64
OR	80	12.0282	12.4314	4.28	219.98	51.40
OR	90	12.1528	12.3448	4.5	242.55	53.90
OR	100	12.0969	12.5072	4.72	265.4	56.23
IMPLY	1	27.3601	26.7455	1.16	1.15	0.99
IMPLY	10	27.0111	26.2724	1.38	11.7	8.48
IMPLY	20	26.2047	26.8411	1.6	23.34	14.59
IMPLY	30	25.9717	26.8567	1.82	35.1	19.29
IMPLY	40	25.442	26.962	2.04	46.74	22.91
IMPLY	50	25.2755	27.2387	2.26	58.6	25.93
IMPLY	60	25.5642	26.6399	2.48	71.35	28.77
IMPLY	70	25.696	27.795	2.75	84.04	30.56
IMPLY	80	25.072	27.4602	2.97	96.67	32.55
IMPLY	90	25.3899	27.0263	3.19	109.36	34.28
IMPLY	100	25.4739	26.9153	3.41	122.1	35.81
AOI	3	17.46849	17.29321	5.83	5.33	0.91423671
AOI	30	17.25798	17.23958	6.38	69.04	10.8213166
AOI	60	17.0877	17.38741	6.93	152.63	22.024531
AOI	90	16.85221	17.40994	7.43	216.07	29.0807537
AOI	120	16.94005	17.52239	8.03	276.44	34.4259029
AOI	150	16.83608	17.45485	8.52	334.93	39.3110329
AOI	180	16.92008	17.38057	9.07	394.76	43.5237045
AOI	210	16.84946	17.75459	9.61	455.17	47.364204
AOI	240	16.5152	17.53659	10.21	514	50.3428012
AOI	270	16.73702	17.36748	10.76	573.59	53.3076208
AOI	300	16.52319	17.32919	11.37	633.29	55.6983280

18, 22, 35, 40, 42, 47, 50], and, as the time progresses, the best access time found up to certain time units is measured. The function is therefore decreasing. In case of AND, both algorithms started from value about 33. PHC algorithm dropped the access time below 28.5 after 20 seconds, and reached apparent optimum 28.4 after 40 seconds, with no further improvement. RP algorithm [5], however, dropped below 32 access time after 600 seconds, and after 1800 seconds it reached only 31.9 access time (the 1800 was time limit set). In case of OR function, access time started about 19 for both algorithms. It dropped below 18 after 20 seconds in PHC algorithm, reaching apparent optimum 17.9 time units after 70 seconds. In case of RP algorithm [5], the access time

dropped below 18.65 after 600 seconds, and did not improve by the expiration time 1800 seconds. In case of IMPLY problem, access time started at about 47 time units. For PHC algorithm, it dropped below 43 after 40 seconds, and after 100 seconds it reached apparent optimum at 42.8. After 200 seconds the access time dropped below 46.4 for RP algorithm [5], and after 1800 seconds it was still above 46.2. Figure 1 shows the access times of two algorithms for IMPLY problem, as a function of processing time. It is expected that RP algorithm needs more than hundred times more time to reach optimal solution than HPC algorithm, that is, in 4000–10000 seconds range. The combined AOI problem has showed similar comparison (it appears that its performance is close to linear combination

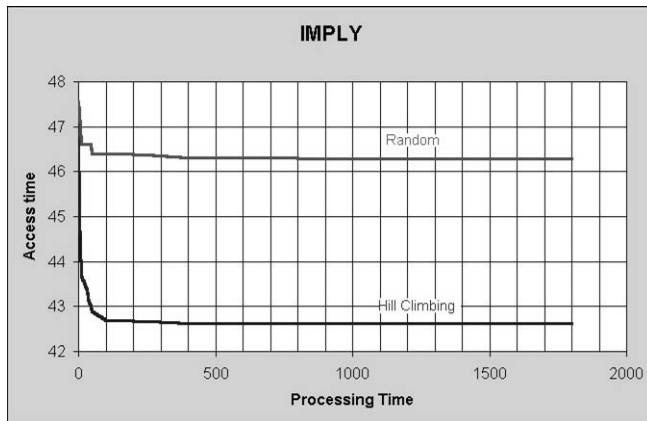


Figure 1. Access time for the RP and PHC algorithms for the IMPLY problem, $n = 50$, $t = 10$ are file grouping [7, 18, 22, 35, 40, 42, 47, 50].

of performances of its ingredients). Note that the times appear to be relatively high, which is due to slower processing in Matlab as compared to processing in, say, C or C++.

The third set of experiments was designed to compare the times needed to evaluate certain number of permutations by both algorithms. The value $n = 30$ is selected, with grouping [2, 7, 10, 15, 18, 20, 24, 26, 30]. As expected, RP algorithm requires approximately n time more time to evaluate m permutations with respect to evaluating one permutation. On the other hand, hill climbing algorithm needs similar time to evaluate the first permutation, but does not even double that time until it evaluates hundred more permutations for AND and OR cases, and approximately doubles for AOI problem, and triples the time for IMPLY case. Also, the access time for hill climbing was always lower, after the same number of evaluated permutations, except sometimes after first few permutations. The detailed results are given in table 2. The access times are not strictly decreasing since the program was run again for each number of permutations (that is, it was not extended by evaluating additional permutations, but was started again, for better reliability of the comparison). The ratio of processing times increases with number of permutations evaluated, and is over $n = 30$ after 30, 30, and 70 permutations, respectively (the ratios are shown in the last column).

6. Conclusions

It appears from our experimental results that there is clear bound on the access time that can be achieved. That is, both parallel hill climbing and random permutations algorithms are able to arrive close to the optimal access time, but they need different computational time to achieve such quality of best solution. Hill climbing algorithm is $O(n)$ times faster, where n is the number of files to be broadcast. In other words, the solution space, although huge, appears to be quite homogeneous, and a “nice” cruise though it is possible. Based on that observation, it appears also that no significantly faster algorithm, achieving same quality, can be described by applying more sophisticated and generally considered more effective methods, such as genetic algorithms, tabu search or

simulated annealing. The optimal solution is reachable from many starting points, thus it is more important to provide fast “climbing” for each given “hill” than sophisticated type of climbing, since there are sufficient number of hills that turn into “mountains”, that is, have near optimal access times at their tops. The fast climbing in our paper is achieved by reducing the evaluation time of most permutations from $O(n^2)$ to $O(n)$. It is achieved by using already evaluated access time for the update on access time of any neighboring permutation, which is obtained by swapping two files in it.

In this paper, and in [5], it is assumed that the servers know only pairwise dependencies. In the more general case the servers could accumulate dependencies between three or more files. Also, it was assumed that each provider receives the same share. In a more general environment, some providers may get more than proportional time for their files. These extensions remain an open problem for further research.

Acknowledgements

This work was partially supported by NSERC and CONACYT-Mexico (37017-A) research grants.

References

- [1] S. Acharya, M.J. Franklin and S. Zdonik, Dissemination-based data delivery using broadcast disks, *IEEE Personal Communications* 2(6) (1995) 50–60.
- [2] D.H. Ackley, *A Connectionist Machine for Genetic Hill Climbing* (Kluwer, 1987).
- [3] D. Aksoy and M.J. Franklin, A scheduling approach for large scale on-demand data broadcast, *IEEE/ACM Trans. Networking* 7(6) (1999) 846–860.
- [4] M.H. Ammar and J.W. Wong, On the optimality of cyclic transmission in Teletext systems, *IEEE Trans. Communications* 35(1) (1987) 68–73.
- [5] A. Bar-Noy, J. Naor and B. Schieber, Pushing dependent data in client-providers-servers systems, in: *Proc. ACM Mobile Computing Conference, MOBICOM 2000* (2000) pp. 222–230.
- [6] T.C. Belding, Potholes on the royal road, in: *Proc. of the Genetic and Evolutionary Computation Conf. GECCO*, San Francisco, CA (2001) pp. 211–218.
- [7] A. Datta, D.E. VanderMeer, A. Celik and V. Kumar, Broadcast protocols to support efficient retrieval from databases by mobile users, *ACM Trans. Database Systems (TODS)* 24(2) (1999) 1–79.
- [8] T. Imielinski, S. Viswanathan and B. Badrinath, Energy efficient indexing on air, in: *Proc. ACM SIGMOD Internat. Conf. Management of Data* (1994) pp. 25–36.
- [9] C. Kenyon and N. Schabanel, The data broadcast problem with non-uniform transmission times, in: *Proc. ACM-SIAM Sympos. Discrete Algorithms SODA* (1999) pp. 547–556.
- [10] W.C. Lee, Q.L. Hu and D.L. Lee, A study of channel allocation methods for data dissemination in mobile computing environments, *ACM/Baltzer Journal of Mobile Networks and Applications (MONET)* 4(2) (1999) 117–129.
- [11] W.C. Lee and D.L. Lee, Signature caching techniques for information broadcast and filtering in mobile environments, *ACM/Baltzer Journal of Wireless Networks (WINET)* 5(1) (1999) 57–67.
- [12] C.W. Lin and D.L. Lee, Adaptive data delivery in wireless communication environments, in: *Proc. IEEE Internat. Conf. Distributed Computing Systems ICDCS*, Taiwan (April 2000) pp. 444–452.

- [13] N.J. Nilsson, *Artificial Intelligence, A New Synthesis* (Morgan Kaufmann, 1998).
- [14] W.C. Peng and M.S. Chen, Dynamic generation of data broadcasting programs for a broadcast disk array in a mobile computing environment, in: *Proc. ACM Internat. Conf. Information and Knowledge Management*, McLean, VA, USA (November 2000) pp. 38–45.
- [15] S. Russel and P. Norvig, *Artificial Intelligence, A Modern Approach* (Prentice-Hall, 1995).
- [16] J. Xu, Q.L. Hu, D.L. Lee and W.C. Lee, SAIU: An efficient cache replacement policy for wireless on-demand broadcast, in: *Proc. ACM Internat. Conf. Information Knowledge Management*, McLean, VA, USA (November 2000) pp. 46–53.
- [17] J. Xu, D.-L. Lee, Q. Hu and W.-C. Lee, Data broadcast, in: *Handbook of Wireless Networks and Mobile Computing*, ed. I. Stojmenović (Wiley, 2002).



Francisco J. Ovalle-Martínez was born in Mexico City in 1977. He received his bachelor's degree in communications engineering (signal processing) from the School of Engineering UNAM-Mexico in 2001. He has worked as a Junior Programmer at Pegasus Control, a software and hardware development firm in Mexico. Currently, he holds a Conacyt-Mexico grant to pursue M.Sc. studies in Computer Science at the University of Ottawa. His interest areas include mobile computing, digital communications, computer networks and networks security.

E-mail: Fovalle@site.uottawa.ca



Julio Solano González received his degree in electrical engineering from the Autonomous National University of Mexico (UNAM), in 1984. He received his I.P. Diploma in 1985 from the Philips International Institute of Technological Studies, Eindhoven, The Netherlands, and his Ph.D. degree in parallel computer systems engineering from the University of Wales, Great Britain in 1992. During the same year, he joined the Institute of Research in Applied Mathematics and Systems (IIMAS) at UNAM. He is currently Associate Professor and Head of the Computer Systems Engineer-

ing and Automation Department at IIMAS-UNAM. His research interests include high performance computer algorithms and architectures, evolutionary computation for signal and image processing and wireless networks.
E-mail: julio@uxdea4.iimas.unam.mx



Ivan Stojmenović received the B.S. and M.S. degrees in 1979 and 1983, respectively, from the University of Novi Sad and Ph.D. degree in mathematics in 1985 from the University of Zagreb. He earned a third degree prize at International Mathematics Olympiad for high school students in 1976. In 1980 he joined the Institute of Mathematics, University of Novi Sad (Yugoslavia). In Fall 1988, he joined the faculty in the Computer Science Department at the University of Ottawa (Canada), where currently he

holds the position of a Full Professor and coordinator of graduate programs in computer science in SITE. He has published three books and over 150 different papers in journals and conferences. His research interests are wireless networks, parallel computing, multiple-valued logic, evolutionary computing, neural networks, combinatorial algorithms, computational geometry and graph theory. He is currently a managing editor of Multiple-Valued Logic and Soft Computing, and an editor of the following journals: IEEE Transactions on Parallel and Distributed Computing, Parallel Processing Letters, Parallel Algorithms and Applications, and International Journal of Computers and Applications.

E-mail: stojmenovic@direcway.com