Wireless Netw (2008) 14:219–231
DOI 10.1007/s11276-006-9146-x

Efficient heuristics for data broadcasting on multiple channels

S. Anticaglia · F. Barsi · A.A. Bertossi · L. Iamele · M.C. Pinotti

Published online: 9 October 2006

Abstract The problem of data broadcasting over multiple channels consists in partitioning data among channels, depending on data popularities, and then cyclically transmitting them over each channel so that the average waiting time of the clients is minimized. Such a problem is known to be polynomially time solvable for uniform length data items, while it is computationally intractable for non-uniform length data items. In this paper, two new heuristics are proposed which exploit a novel characterization of optimal solutions for the special case of two channels and data items of uniform lengths. Sub-optimal solutions for the most general case of an arbitrary number of channels and data items of non-uniform lengths are provided. The first heuristic, called *Greedy+*, combines the novel characterization with the known greedy approach, while the second heuristic, called *Dlinear*, combines the same characterization with the dynamic programming technique. Such heuristics have been tested on benchmarks whose popularities are characterized by Zipf distributions, as well as on a wider set of benchmarks. The experimental tests reveal that Dlinear finds optimal solutions almost always, requiring good running times. However, Greedy+ is faster and scales well when changes occur on the input parameters, but provides solutions which are close to the optimum.

This work has been supported by ISTI-CNR under the BREW research grant.

S. Anticaglia · F. Barsi · L. Iamele · M.C. Pinotti (✉)
Department of Computer Science and Mathematics, University of Perugia, 06123 Perugia, Italy
e-mail: pinotti@unipg.it

A.A. Bertossi
Department of Computer Science, University of Bologna, 40127 Bologna, Italy

Keywords Wireless communication · Data broadcasting · Multiple channels · Heuristics · Flat scheduling · Average waiting time · Dynamic programming

1 Introduction

Broadcasting is an efficient way of simultaneously disseminating data to a large number of clients. It is especially effective in an asymmetric wireless environment, where a server at a base-station repeatedly transmits data items from a given set over multiple wireless channels, while clients wait for their desired item on the proper channel [1, 3, 15, 16].

As applications, consider data services broadcast by base-stations of cellular networks or hot-spot areas, such as stock quotes, weather info, traffic news, video clips, movie trailers, sport scores, transport time tables [7]. In these applications, as the time spent for mobile clients passing through the base-station coverage areas is very short, it is of paramount importance to reduce the waiting time of the clients.

The client expected delay increases with the size of the set of the data items to be transmitted by the server. Indeed, the client has to wait for many unwanted data before receiving his own data. In a multi-channel environment, an allocation strategy has to be pursued so as to assign data items to channels. Moreover, each client can access either only a single channel or any available channel at a time. In the former case, if the client can access only one prefixed channel and can potentially retrieve any available data, then all data items must be replicated over all channels. Otherwise, data can be partitioned among the channels, thus assigning each item to only one channel. In this latter case, the efficiency can be improved by adding an index that informs the client at which time and on which channel the desired item will be transmitted. In this way, the mobile client can save battery energy because, after reading the index info, it can sleep and wake up on the proper channel just before the transmission of the desired item.

Several variants for the problem of data allocation and broadcast scheduling have been proposed in the literature, which depend on the perspectives faced by the research communities [2, 3, 5, 9–11, 13, 14, 16–18].

Specifically, the networking community faces a version of the problem, known as the *Broadcast Problem*, whose goal is to find an infinite schedule on a single channel [5, 10, 11, 16]. Such a problem was first introduced in the teletext systems by [2, 3]. Although it is widely studied (e.g., it can be modeled as a special case of the Maintenance Scheduling Problem and the Multi-Item Replenishment Problem [5, 10]), its tractability is still under consideration. Therefore, the emphasis is on finding near optimal schedules for a single channel. Almost all the proposed solutions follow the *square root rule (SRR)* [3]. The aim of such a rule is to produce a broadcast schedule where each data item appears with equally spaced replicas, whose frequency is proportional to the square root of its popularity and inversely proportional to the square root of its length. The multi-channel schedule is obtained by distributing in a round robin fashion the schedule for a single channel [16]. Since each item appears in multiple replicas which, in practice, are not equally spaced, these solutions make indexing techniques not effective. Briefly, the main results known in the literature for the Broadcast Problem can be summarized as follows. For *uniform* lengths, namely all items of the same length, it is still unknown whether the problem can be solved in polynomial time or not. For a constant number of channels, the best algorithm proposed so far is the Polynomial Time Approximation Scheme (PTAS) devised in [11]. In contrast, for *non-uniform* lengths, the problem has been shown to be strong *NP*-hard even for a single channel, a 3-approximation algorithm was devised for one channel, and a heuristic has been proposed for multiple channels [10].

On the other hand, the database community seeks for a periodic broadcast scheduling which should be easily indexed [9]. For the single channel, the obvious schedule that admits index is the *flat* one. It consists in selecting an order among the data items, and then transmitting them once at a time, in a round-robin fashion [1], producing an infinite periodic schedule. In a flat schedule indexing is trivial, since each item will appear once, and exactly at the same relative time, within each period. Although indexing allows the client to sleep and save battery energy, the client expected delay is half of the schedule period and can become infeasible for a large period. To decrease the client expected delay, still preserving indexing, flat schedules on multiple channels can be adopted [13, 14, 18]. However, in such a case the allocation of data to channels becomes critical. For example, allocating items in a balanced way simply scales the expected delay by a factor equal to the number of channels. To overcome this drawback, *skewed* allocations have been proposed where items are partitioned according to their popularities so that the most requested items appear in a channel with shorter period [13, 18]. Hence, the resulting problem is slightly different from the Broadcast Problem since, in order to minimize the client expected delay, it assumes skewed allocation and flat scheduling. This variant of the problem is easier than the Broadcast Problem. Indeed, the problem has been shown to be polynomial time solvable for uniform length data items [18], and it has been proved to be computationally intractable (*NP*-hard) for non-uniform length data items [4].

The present paper expands the work started in [4, 18] for the problem of data broadcasting over multiple channels, with the objective of minimizing the average waiting time of the clients, under the assumptions of skewed allocation to channels and flat scheduling per channel. In [4, 18], several algorithms have been proposed, all of which assume that a sorting preprocessing step has been done on the data items. In the uniform case, the fastest known algorithm producing an optimal solution requires $O(NK \log N)$ time [4], where N is the number of items and K is the number of channels. Such an optimal algorithm is based on dynamic programming and solves NK problem instances, for $1 \leq n \leq N$ and $1 \leq k \leq K$. In the non-uniform case, the problem can be optimally solved in pseudo-polynomial time when $K = 2$, by a reduction to a knapsack problem, and in exponential time for arbitrary K [4]. In this latter case, a heuristic, called *Greedy*, has been proposed in [18]. Fixed N, Greedy starts with all data items assigned to one channel, and then proceeds by splitting the items of one channel between two channels, thus adding a new channel, until K channels are reached. In practice, Greedy is very fast, scales with the number of channels, and provides fair sub-optimal solutions for the K instances of the problem, where N is fixed and $1 \leq k \leq K$.

This paper presents two new heuristics which provide sub-optimal solutions for the data broadcasting problem with non-uniform data item lengths and an arbitrary number of channels. As with Greedy, both heuristics assume that the items are sorted by decreasing popularities per length unit. As opposed to Greedy, they pretend that a characterization of the optimal solution of the problem for $K = 2$ and uniform lengths holds also for the general case of arbitrary K and non-uniform lengths. The first heuristic, called *Greedy*+, follows the same strategy as Greedy. To solve K instances with N fixed and $1 \leq k \leq K$, it requires $O(NK)$ time in the worst case, The second heuristic, called *Dlinear*, follows the dynamic programming relation proposed in [18] and requires an $O(NK)$ time for solving all the NK instances, for $1 \leq n \leq N$ and $1 \leq k \leq K$. The proposed heuristics are experimentally tested on some benchmarks, whose popularities are characterized by Zipf and Stairs distributions. The Zipf distribution has been shown to characterize the popularity of one element among a set of similar data, like a web page in a web site [6], while the Stairs distribution is employed because it models bad instances for the proposed heuristics. The experimental tests reveal that the quality of the solution provided by Dlinear is much better than that produced by the other heuristics. Indeed, Dlinear finds optimal solutions almost always, requiring reasonable running times. Although Greedy remains the fastest heuristic, it gives the worst sub-optimal solutions. Both the running times and the quality of the solutions of Greedy+ are intermediate between those of Dlinear and Greedy. As Greedy, Greedy+ scales well with respect to all parameters changes.

The rest of this paper is so organized. Section 2 gives notations, definitions and the problem statement. In particular, Section 2.1 proves the novel characterization of the optimal solution, for the special case with $K = 2$ and uniform lengths, that motivates the new heuristics. Section 3 presents the $O(NK)$ time Greedy+ and Dlinear heuristics. Moreover, it is also shown that the Greedy+ algorithm, when restricted to uniform length data, can be implemented so as to take $O(K \log N)$ time in the worst case. Section 4 reports the experimental tests of the heuristics, performed on randomly generated instances. Conclusions are offered in Section 5. Finally, the Appendix shows that the worst case time complexity of the original Greedy algorithm given in [18] is $O(NK)$, even for uniform length data, while its average time complexity is $O(N \log K)$.

2 Preliminaries

Consider a set of K identical channels, and a set $D = \{d_1, d_2, \ldots, d_N\}$ of N data items. Each item d_i is characterized by a *popularity* p_i and a *length* z_i, with $1 \leq i \leq N$. The popularity p_i represents how frequently item d_i is requested by the clients, and it does not vary along the time. Popularities can be either arbitrary positive integers, or real numbers normalized in the range (0, 1] such that $\sum_{i=1}^{N} p_i = 1$. The length z_i is an integer number, counting how many time units (or, ticks) are required to transmit item d_i on any channel. When all data lengths are the same, i.e. $z_i = z$ for $1 \leq i \leq N$, the lengths are called *uniform* and are assumed to be unit, i.e. $z = 1$. When the data lengths are not the same, the lengths are said *non-uniform*.

The items have to be partitioned into K groups $G_1, \ldots, G_K$. Group G_j collects the data items assigned to channel j, with $1 \leq j \leq K$. The cardinality of G_j is denoted by N_j, the sum of its item lengths is denoted by Z_j, i.e. $Z_j = \sum_{d_i \in G_j} z_i$, and the sum of its popularities is denoted by P_j, i.e. $P_j = \sum_{d_i \in G_j} p_i$. Note that since the items in G_j are cyclically broadcast according to a flat schedule, Z_j is the schedule period on channel j. Clearly, in the uniform case $Z_j = N_j$, for $1 \leq j \leq K$. If item d_i is assigned to channel j, and assuming that clients can start to listen at any instant of time with the same probability, the *client expected delay* for receiving item d_i is half of the period, namely $\frac{Z_j}{2}$. Assuming, as in [18], that indexing allows clients to know in advance the content of the channels, the *average expected delay* (AED) over all channels is

$$\text{AED} = \frac{1}{2} \sum_{j=1}^{K} Z_j P_j \tag{1}$$

Given K channels, a set D of N items, where each data item d_i comes along with its popularity p_i and its integer length z_i, the data broadcasting problem consists in partitioning D into K groups $G_1, \ldots, G_K$, so as to minimize the objective function AED given in Eq. (1). In the special case of equal lengths, the corresponding objective function is derived replacing Z_j with N_j in Eq. (1).

Some known results, proposed in [4, 18], that will be used in the next sections, are now briefly recalled.

Lemma 1 ([18]). *Let G_h and G_j be two groups in an optimal solution for a problem instance with uniform lengths. Let d_i and d_k be items with $d_i \in G_h$ and $d_k \in G_j$. If $N_h < N_j$, then $p_i \geq p_k$. Similarly, if $p_i > p_k$, then $N_h \leq N_j$.*

In other words, the most popular items are allocated to less loaded channels so that they appear more frequently. As a consequence, if the items are sorted by non-increasing popularities, then the group sizes are non-decreasing.

Corollary 1 ([18]). *Let $d_1, d_2, \ldots, d_N$ be N uniform length items with $p_i \geq p_k$ whenever $i < k$. Then, there exists an optimal solution for partitioning them into K groups $G_1, \ldots, G_K$, where each group is made of consecutive elements.*

By the above corollary, in the uniform case the items are assumed to be sorted by non-increasing popularities, and any solution S will be compactly represented by a *segmentation*, that is a $(K-1)$-tuple $(B_1, B_2, \ldots, B_{K-1})$, where B_j is the index, called *border*, of the rightmost item belonging to group G_j and $B_1 < B_2 < \cdots < B_{K-1}$. Notice that the cardinality of G_j, i.e. the number N_j of items in the group, is $N_j = B_j - B_{j-1}$, where $B_0 = 0$ and $B_K = N$ are assumed. From now on, a segmentation $S = (B_1, B_2, \ldots, B_{K-1})$ for the uniform case is called *feasible* if $N_1 \leq N_2 \leq \cdots \leq N_K$. Indeed, by Lemma 1, an optimal solution will be sought only among feasible solutions.

For any $n \leq N$ and $k \leq K$, let $opt_{n,k}$ denote the cost of an optimal solution for items $d_1, \ldots d_n$ and k channels (groups). Let $C_{i,h}$ be the cost of assigning consecutive items $d_i, \ldots, d_h$

to one group, i.e. $C_{i,h} = \frac{1}{2}(h-i+1)\sum_{q=i}^{h} p_q$. The following result holds.

Theorem 1 ([18]). *Let $d_1, d_2, \dots, d_N$ be N uniform length items, sorted by non-increasing popularities. Hence,*

$$opt_{n,k} = \begin{cases} C_{1,n} & \text{if } k = 1 \\ \min_{1\le \ell \le n-1}\{opt_{\ell,k-1} + C_{\ell+1,n}\} & \text{if } k > 1 \end{cases} \tag{2}$$

Theorem 1 suggests an $O(N^2K)$ time dynamic programming algorithm to solve the problem in the uniform case. Indeed, consider the $K \times N$ matrix M with $M_{k,n} = opt_{n,k}$. The entries of M are computed row by row applying Recurrence 2. In order to actually construct an optimal partition, a second matrix F is employed which stores in $F_{k,n}$ the value of ℓ which minimizes the right-hand-side of Eq. (2). Hence, the optimal solution for N and K is given by $S = (B_1, B_2, \dots, B_{K-1})$ where, starting from $B_K = N$, the value of B_k is equal to $F_{k+1,B_{k+1}}$, for $k = K-1, \dots, 1$. A useful property of the optimal solution is that the values stored in each row of matrix F are non-decreasing, as stated in the following Lemma:

Lemma 2 ([4]). *Let $d_1, d_2, \dots, d_N$ be N uniform length items, sorted by non-increasing popularities. For any $n \le N$ and $k \le K$, $F_{k,n-1} \le F_{k,n}$.*

In words, Lemma 2 implies that, given the items sorted by non-increasing popularities, if one builds an optimal solution for N items from an optimal solution for $N-1$ items, then the border B_{K-1} can only move to the right.

2.1 Two channels and uniform lengths

This subsection exploits the structure of the optimal solution in the special case where the item lengths are uniform and there are only two channels. The problem is thus to find a partition S into G_1 and G_2 such that $AED_S = \frac{1}{2}(N_1P_1 + N_2P_2)$ is minimized. Clearly, $N = N_1 + N_2$, and by Lemma 1, $N_1 \le N_2$ holds for any optimal solution. Moreover, any feasible solution S can be denoted by the single border B_1, which coincides with N_1.

Lemma 3. *Consider N uniform length items, sorted by non-increasing popularities, and $K = 2$ channels. Let $S = (N_1)$ be a feasible solution such that $P_1 \le P_2$. If the solution $S' = (N_1 + 1)$ is feasible, then $AED_{S'} \le AED_S$.*

Proof: Since S' is feasible, then $N_1 + 1 \le N_2 - 1$. The new solution S' differs from S because item d_{N_1+1} is moved from G_2 to G_1. Therefore, $AED_{S'} = \frac{1}{2}((N_1+1)(P_1 + p_{N_1+1}) + (N_2-1)(P_2 - p_{N_2-1})) = \frac{1}{2}(N_1P_1 + N_2P_2 + (N_1 - N_2 + 2)p_{N_1+1} + (P_1 - P_2))$.

Since $AED_S = \frac{1}{2}(N_1P_1 + N_2P_2)$, $N_1 - N_2 + 2 \le 0$, and $P_1 - P_2 \le 0$, it follows that $AED_{S'} \le AED_S$. □

While Lemma 1 gives the upper bound $N_1 \le \lfloor \frac{N}{2} \rfloor$ on the cardinality of group G_1, Lemma 3 provides a lower bound b on N_1. Indeed, Recurrence 2 for $K = 2$ can be rewritten as follows:

$$opt_{N,2} = \min_{b \le \ell \le \lfloor \frac{N}{2} \rfloor} \{C_{1,\ell} + C_{\ell+1,N}\} \tag{3}$$

where

$$b = \max_{1\le s\le \lfloor \frac{N}{2} \rfloor} \left\{ s \; : \; \sum_{h=1}^{s} p_h \le \sum_{h=s+1}^{N} p_h \right\}.$$

The following lemma improves on the upper bound of N_1 given by Lemma 1, and shows that the values of the feasible solutions assumed in the right-hand side of Eq. (3) form a *unimodal* sequence, namely there is a particular index ℓ such that the values on its left are in non-increasing order, while those on its right are in increasing order.

Lemma 4. *Consider N uniform length items, sorted by non-increasing popularities, and $K = 2$ channels. Let $S = (N_1)$ be a feasible solution such that $P_1 > P_2$. Consider the solutions $S' = (N_1 + 1)$ and $S'' = (N_1 + 2)$. If $AED_{S'} > AED_S$, then $AED_{S''} > AED_{S'}$.*

Proof: By definition, $AED_S = \frac{1}{2}(N_1P_1 + N_2P_2)$ and $AED_{S'} = \frac{1}{2}((N_1+1)(P_1 + p_{N_1+1}) + (N_2-1)(P_2 - p_{N_1+1})) = AED_S + \frac{1}{2}((P_1 - P_2) + p_{N_1+1}(N_1 - N_2 + 2))$.

Since $AED_{S'} > AED_S$, it follows that $(P_1 - P_2) > p_{N_1+1}(N_2 - N_1 - 2)$.

Moreover, $AED_{S''} = \frac{1}{2}(N_1+2)(P_1 + p_{N_1+1} + p_{N_1+2}) + \frac{1}{2}(N_2-2)(P_2 - p_{N_1+1} - p_{N_1+2})$, and thus $AED_{S''} - AED_{S'} = \frac{1}{2}(P_1 - P_2) + \frac{1}{2}p_{N_1+2}(N_1 - N_2 + 2) + (p_{N_1+1} + p_{N_1+2})$.

Since $p_{N_1+1} \ge p_{N_1+2}$, one has $(P_1 - P_2) > p_{N_1+2}(N_2 - N_1 - 2)$.

Finally, $AED_{S''} > AED_{S'}$ holds because $\frac{1}{2}(P_1 - P_2) + \frac{1}{2}p_{N_1+2}(N_1 - N_2 + 2) + (p_{N_1+1} + p_{N_1+2}) > \frac{1}{2}(P_1 - P_2) + \frac{1}{2}p_{N_1+2}(N_1 - N_2 + 2) > 0$. □

Let $f(\ell) = C_{1,\ell} + C_{\ell+1,N} = \frac{\ell}{2}\sum_{h=1}^{\ell} p_h + \frac{N-\ell}{2}\sum_{h=\ell+1}^{N} p_h$. Then, the border m that minimizes Eq. (3), that is the optimal solution of the problem, is given by:

$$opt_{N,2} = f(m) \tag{4}$$

```
Procedure BinSearch $(i, j)$;
    $m \leftarrow \lfloor \frac{i+j}{2} \rfloor$;
    if $i = j$ then
        return $m$
    else
        if $f(m) \geq f(m+1)$ then
            BinSearch $(m+1, j)$
        else
            BinSearch $(i, m)$;
```

Fig. 1 The binary search on a unimodal sequence

where

$$m = \min_{b \leq \ell \leq \lfloor \frac{N}{2} \rfloor} \{\ell \,:\, f(\ell) < f(\ell+1)\}\,.$$

Due to the unimodal property of the sequence of values $f(b), f(b+1), \ldots, f(\lfloor \frac{N}{2} \rfloor)$, the search of m can be done in $O(\log N)$ time by a suitable modified binary search [8], as shown in Fig. 1.

3 New heuristics

The purpose of the new heuristics to be presented in this section is to quickly find good sub-optimal solutions for the most general case of non-uniform data lengths and an arbitrary number of channels. Such a goal is achieved by pretending that the optimal solution characterization, proved in Section 2.1 for the special case of two channels and uniform lengths, holds also in the general case of more than two channels and non-uniform lengths.

As with all the previously known heuristics, the new heuristics also assume that the items are sorted by non-increasing $\frac{p_i}{z_i}$ ratios. This can be done in $O(N \log N)$ time by a sorting preprocessing step. Moreover, since the lengths are non-uniform, the cost of assigning the items from d_i to d_j to a single channel becomes $C_{i,j} = \frac{1}{2}(\sum_{h=i}^{j} p_h)(\sum_{h=i}^{j} z_h)$. Letting $P_{i,j} = \sum_{h=i}^{j} p_h$ and $Z_{i,j} = \sum_{h=i}^{j} z_h$, one notes that all the $P_{1,n}$ and $Z_{1,n}$, for $1 \leq n \leq N$, can be computed in $O(N)$ time by two prefix sum computations, performed as a preprocessing step. Hence, a single $C_{i,j}$ can be computed on the fly in constant time as $C_{i,j} = \frac{1}{2}(P_{1,j} - P_{1,i-1})(Z_{1,j} - Z_{1,i-1})$. From now on, in order to simplify the presentation, $C_{i,j}$ is defined to be 0 whenever $i > j$.

Since all the heuristics assume the two above preprocessing steps, their time complexity will not be included in the complexity analysis of the heuristics.

3.1 The Greedy+ algorithm

The Greedy+ heuristic is a refinement of the Greedy heuristic presented in [17]. Recall that the Greedy heuristic works for a fixed number N of data items. It initially assigns all the N items to a single group. Then, for $K - 1$ times, one of the groups is split in two groups, that will be assigned to two different channels. To find which group to split along with its actual split point, all the possible points of all groups are considered as split point candidates, and the one that decreases AED the most is selected. An efficient implementation takes advantage from the fact that, between two subsequent splits, it is sufficient to recompute the costs for the split point candidates of the last group that has been actually split.

In summary, Greedy+ consists of two phases. In the first phase it behaves as Greedy, except for the way the split point is determined. In the second phase, the solution provided by the first phase is refined by working on pairs of consecutive channels.

Specifically, in the first phase, Greedy+ uses an approach similar to that of Eq. (4) to determine the split point. This is because splitting one channel is the same as solving the data broadcast problem for two channels. In details, assume that the channel to be split contains the items from d_i to d_j, with $1 \leq i < j \leq N$, and let $cost_{i,j,2}$ denote the cost of a feasible solution for assigning such items to two channels. Then, the split point is given by the value of m that satisfies the following relation:

$$cost_{i,j,2} = C_{i,m} + C_{m+1,j} \tag{5}$$

where

$$m = \min_{i \leq \ell \leq j-1} \{\ell \,:\, C_{i,\ell} + C_{\ell+1,j} < C_{i,\ell+1} + C_{\ell+2,j}\}.$$

Note that, since the item lengths are not uniform, the sequence of values $C_{i,\ell} + C_{\ell+1,j}$, for $i \leq \ell \leq j-1$, is not unimodal. However, Greedy+ behaves as such a sequence were unimodal. Hence, instead of trying all the possible values of ℓ between i and j, as done by Greedy, Greedy+ performs a left-to-right scan starting from i and stopping as soon the AED increases. In this way, a sub-optimal solution $S = (B_1, B_2, \ldots. B_{K-1})$ is found.

The second phase is performed only when $K \geq 3$ and consists in refining the solution S by recomputing its borders. It consists in a sequence of odd steps, followed by a sequence of even steps. During the t-th odd step, $1 \leq t \leq \lfloor \frac{K}{2} \rfloor$, the two-channel subproblem including the items assigned to groups G_{2t-1} and G_{2t} is solved. Specifically, Eq. (5) is applied choosing $i = B_{2t-2} + 1$ and $j = B_{2t}$, thus recomputing the border B_{2t-1} of S. Similarly, during the t-th even step, $1 \leq t \leq \lfloor \frac{K-1}{2} \rfloor$, the two-channel subproblem including the items assigned to groups G_{2t} and G_{2t+1} is solved by applying Eq. (5) with $i = B_{2t-1} + 1$ and $j = B_{2t+1}$, thus recomputing the border B_{2t} of S.

```
Procedure Greedy+($\{d_1, d_2, \ldots, d_N\}, K$);
    CreateEmptyHeap ($H$);
    Split ($1, N, m, \Delta$);
    InsertHeap ($H, 1, N, m, \Delta$);
    Greedy ($1, K$);
    if $K \geq 3$ then
        for $t$ from 1 to $\lfloor \frac{K}{2} \rfloor$ do
            Split ($B_{2t-2} + 1, B_{2t}, s, \Delta$);
            $B_{2t-1} \leftarrow s$;
        for $t$ from 1 to $\lfloor \frac{K-1}{2} \rfloor$ do
            Split ($B_{2t-1} + 1, B_{2t+1}, s, \Delta$);
            $B_{2t} \leftarrow s$;

Procedure Greedy ($k, K$);
    if $k < K$ then
        DeleteMaxHeap ($H, i, j, m, \Delta$);
        $B_k \leftarrow m$;
        Split ($i, m, s, \Delta$);
        InsertHeap ($H, i, m, s, \Delta$);
        Split ($m + 1, j, s, \Delta$);
        InsertHeap ($H, m + 1, j, s, \Delta$);
        Greedy ($k + 1, K$);
    else
        Sort ($B_1, \ldots, B_{K-1}$);

Procedure Split ($i, j, m, \Delta$);
    $\ell \leftarrow i$;
    $m \leftarrow i$;
    $f \leftarrow C_{i,\ell} + C_{\ell+1,j}$;
    $incr$ ← false;
    while $\ell \leq j - 2$ and $\neg$ $incr$ do
        $temp \leftarrow C_{i,\ell+1} + C_{\ell+2,j}$;
        if $f \geq temp$ then
            $\ell \leftarrow \ell + 1$ ;
            $f \leftarrow temp$
        else
            $incr$ ← true;
    $m \leftarrow \ell$;
    $\Delta \leftarrow C_{i,j} - f$;
```

Fig. 2 The Greedy+ heuristic

The pseudo-code of the *Greedy*+ procedure is depicted in Fig. 2. The first phase of Greedy+ is based on the *Greedy* procedure, which makes use of a heap data structure. Consider a group built so far containing items from d_i to d_j. For such a group, the heap H stores the following four data: the index i of its leftmost item, the index j of its rightmost item, its best split point m, and its AED gain $\Delta = C_{i,j} - (C_{i,m} + C_{m+1,j})$ achieved if the group is split in two groups at m. H stores in its root the group with the largest AED gain Δ. Initially, the data of the single group containing all the items from d_1 to d_N are inserted in H. At invocation k of Greedy, the kth group to be split is found by executing a *DeleteMaxHeap* operation on H, which returns the information stored in the root (namely, i, j, m, Δ), and consequently updates H. The split point m is assigned to the kth border B_k. The *Split* procedure is then invoked once for each of the two new groups containing the items from d_i to d_m and from d_{m+1} to d_j, respectively. Split receives as input the indices of the leftmost and rightmost items of the group, and returns the best split point along with its AED gain. Split iteratively determines the best split point according to Relation 5 (see Fig. 2). The information of the two new groups is then added to H by means of the *InsertHeap* operation. Finally, the borders $B_1, \ldots, B_{k-1}$ are sorted to match the segmentation requirement $B_1 < B_2 < \cdots < B_{K-1}$.

```
Procedure Split ($i, j, m, \Delta$);
    $m \leftarrow$ BinSearch ($i, j$);
    $\Delta \leftarrow C_{i,j} - (C_{i,m} + C_{m+1,j})$;
```

Fig. 3 The Split procedure for uniform lengths

As regard to the time complexity, since H contains at most K items, DeleteMaxHeap and InsertHeap both require $O(\log K)$ time. The final sorting step, executed once, takes $O(K \log K)$ time. Since Split runs in $O(N)$ time, and Greedy is invoked K times, the time complexity of the first phase of Greedy+ is $O(NK)$. The second phase of Greedy+ requires $O(N)$ time since each item is considered as a candidate split point at most in a single Split invocation among all the odd steps, and in a single Split invocation among the even steps. Therefore, the overall time required in the worst case by the Greedy+ heuristic is $O(NK)$, the same as the original Greedy heuristic proposed in [18] (see the Appendix).

In the special case of uniform data lengths, by Lemma 4, the Split procedure merely calls the BinSearch procedure on a unimodal sequence, as shown in Fig. 3. Since the BinSearch procedure, and hence also the Split procedure, takes $O(\log N)$ time, it is easy to see that the worst case time complexity of Greedy+ becomes $O(K \log N)$, improving over the $O(NK)$ time of the original Greedy algorithm [18].

Note that Greedy+ scales well when changes occur on the number of channels, on the number of items, on item popularities, as well as on item lengths. Indeed, adding or removing a channel simply requires doing a new split or removing the last introduced split, respectively. Adding a new item first requires to insert such an item in the sorted item sequence. Assume the new item is added to group G_j, then the border of the two-channel subproblem including items of G_j and G_{j+1} is recomputed by applying Eq. (5). Similarly, deleting an item that belongs to group G_j requires to solve again the two-channel subproblem including items of G_j and G_{j+1}. Finally, a change in the popularity/length of an item is equivalent to first removing that item and then adding the same properly modified item.

3.2 The Dlinear algorithm

The Dlinear heuristic follows a dynamic programming approach similar to that provided by Recurrence 2. It solves all the NK instances, for $1 \leq n \leq N$ and $1 \leq k \leq K$, with the objective of obtaining an $O(NK)$ worst case time complexity. Fixed k, Dlinear computes a solution for n items from the previously computed solution for $n - 1$ items, exploiting the characteristics of the optimal solutions for the uniform case.

For any $n \leq N$ and $k \leq K$, let $M_{k,n}$ denote the cost of a feasible solution for items $d_1, \ldots d_n$ and k channels, and let $F_{k,n}$ be the index of the last element assigned to channel $k-1$ in such a solution. Dlinear selects the feasible solutions that satisfy the following Recurrence:

$$M_{k,n} = \begin{cases} C_{1,n} & \text{if } k = 1 \\ M_{k-1,m} + C_{m+1,n} & \text{if } k > 1 \end{cases} \tag{6}$$

where

$$m = \min_{F_{k,n-1} \leq \ell \leq n-1} \{\ell : M_{k-1,\ell} + C_{\ell+1,n} < M_{k-1,\ell+1} + C_{\ell+2,n}\}.$$

In practice, Dlinear pretends to adapt Recurrence 4, that holds for the uniform data lengths, also to the case of non-uniform data lengths. In particular, the choice of the lower bound $F_{k,n-1}$ in the formula of m is suggested by Lemma 2 which says that the border of channel $k-1$ can only move right when a new item with the smallest popularity is added. Moreover, m is determined as in Eq. (4) pretending that the sequence $M_{k-1,\ell} + C_{\ell+1,n}$, obtained for $F_{k,n-1} \leq \ell \leq n-1$, be unimodal. Therefore, the solution provided by Dlinear is a sub-optimal one.

The pseudo-code for the Dlinear heuristic is shown in Fig. 4. Note that in Loop 1 the leftmost $k-1$ entries in row k of both M and F are meaningless, since at least one element has to be assigned to each channel. The value of m in Recurrence 6 that gives $M_{k,n}$ is computed iteratively in Loop 3 and stored in $F_{k,n}$.

As regard to the time complexity, Loop 3 is performed at most $O(n - F_{k,n-1})$ times. However, such a loop is stopped as soon as *incr* becomes true and hence $F_{k,n} = m$. Therefore, computing $M_{k,n}$ actually requires $O(F_{k,n} - F_{k,n-1})$ time. Hence, computing $M_{k,n}$ for $k+1 \leq n \leq N$ in Loop 2 takes $\sum_{n=k+1}^{N} O(F_{k,n} - F_{k,n-1}) = O(F_{k,N} - F_{k,k}) = O(N)$ time.

```
Input:    $N$ items sorted by non-increasing $\frac{p_i}{z_i}$, and $K$ groups;
Init:         for $n$ from 1 to $N$ do
                  $M_{1,n} \leftarrow C_{1,n}$;
Loop 1:       for $k$ from 2 to $K$ do
                  $F_{k,k} \leftarrow k-1$;
                  $M_{k,k} \leftarrow M_{k-1,k-1} + C_{k,k}$;
Loop 2:           for $n$ from $k+1$ to $N$ do
                      $\ell \leftarrow F_{k,n-1}$;
                      $m \leftarrow \ell$;
                      $M_{k,n} \leftarrow M_{k-1,\ell} + C_{\ell+1,n}$;
                      $incr \leftarrow$ false;
Loop 3:               while $\ell \leq n-2$ and $\neg$ $incr$ do
                          $temp \leftarrow M_{k-1,\ell+1} + C_{\ell+2,n}$;
                          if $M_{k,n} \geq temp$ then
                              $M_{k,n} \leftarrow temp$;
                              $\ell \leftarrow \ell + 1$;
                          else
                              $incr \leftarrow$ true;
                      $m \leftarrow \ell$;
                      $F_{k,n} \leftarrow m$
```

Fig. 4 The Dlinear heuristic

Since Loop 1 is performed $O(K)$ times, the overall time complexity of the Dlinear algorithm is $O(NK)$.

4 Experimental tests

In this section, experimental results, performed on implementations of both the Greedy+ and Dlinear heuristics, are discussed for the data broadcasting problem with K channels and non-uniform lengths. In addition, the implementation of Greedy, as detailed in [17], is used for comparison purposes. The algorithms are written in C and the experiments are run on an AMD Athlon XP 2500+, 1.84 GHz, with 1 GB RAM.

The heuristics are first tested on some non-uniform length instances generated as follows. Given the number N of items and a real number $0 \leq \theta \leq 1$, the item popularities are generated according to a Zipf distribution whose skew is θ, namely:

$$p_i = \frac{(1/i)^\theta}{\sum_{i=1}^{N} (1/i)^\theta} \quad 1 \leq i \leq N$$

In the above formula, $\theta = 0$ stands for a uniform distribution with $p_i = \frac{1}{N}$, while $\theta = 1$ implies a high skew, namely the range of p_i values becomes larger. The item lengths z_i are integers generated according to a uniform distribution in the range $1 \leq z_i \leq z$, as in [16]. The items are sorted by non-increasing $\frac{p_i}{z_i}$ ratios, as suggested in [16]. The parameters N, K, z, and θ vary, respectively, in the ranges: $500 \leq N \leq 2500$, $10 \leq K \leq 500$, $3 \leq z \leq 10$, and $0.5 \leq \theta \leq 1$.

Since the optimal solutions can be found in a reasonable time only for small values of N and z, a *lower bound* on AED is used for large values of N and z. The lower bound for a non-uniform instance is obtained by transforming it into a uniform instance as follows. Each item d_i of popularity p_i and length z_i is decomposed in z_i items of popularity $\frac{p_i}{z_i}$ and length 1. Since more freedom has been introduced, it is clear that the optimal AED for the so transformed problem is a lower bound on the AED of the original problem. Since the transformed problem has uniform lengths, its optimal AED is obtained by running the Dichotomic algorithm presented in [4].

The simulation results are exhibited in Tables 1–4. The tables report the running time, the client AED, and the percentage of error, which is computed as

$$\left(\frac{\text{AED}_{\text{heuristic}} - \text{AED}_{\text{lowerbound}}}{\text{AED}_{\text{lowerbound}}}\right)100$$

The running times reported in the tables do not include the time for sorting, which is the same for all the algorithms. It is worth noting that the algorithm running times are measured in microseconds, while the client AEDs are measured in ticks.

Table 1 Experimental results on Zipf distributions, when $K = 20$, $\theta = 0.8$, and $z = 3$

$N/K/\theta/z$	Algorithm	AED	% Error	Time
500/20/0.8/3	Greedy	18.72	7.1	102
	Greedy +	17.58	0.6	3514
	Dlinear	17.47		2106
	Lower bound	17.47		
1500/20/0.8/3	Greedy	53.85	7.9	283
	Greedy +	51.71	3.6	21240
	Dlinear	49.90		6519
	Lower bound	49.90		
1750/20/0.8/3	Greedy	62.64	7.9	326
	Greedy +	58.92	1.5	31137
	Dlinear	58.04		7488
	Lower bound	58.04		
2000/20/0.8/3	Greedy	71.24	7.9	373
	Greedy +	66.93	1.4	38570
	Dlinear	65.98		8602
	Lower bound	65.98		
2250/20/0.8/3	Greedy	79.70	7.8	457
	Greedy +	75.06	1.6	45170
	Dlinear	73.87		9749
	Lower bound	73.87		
2500/20/0.8/3	Greedy	88.40	7.8	474
	Greedy +	82.51	0.7	62376
	Dlinear	81.93		10920
	Lower bound	81.93		

Table 2 Experimental results on a Zipf distribution, when $N = 2500$, $\theta = 0.8$, and $z = 3$

$N/K/\theta/z$	Algorithm	AED	% Error	Time
2500/10/0.8/3	Greedy	179.16	7.8	381
	Greedy +	167.86	1.0	97356
	Dlinear	166.14		4919
	Lower bound	166.14		
2500/40/0.8/3	Greedy	44.04	7.9	562
	Greedy +	41.58	1.9	34147
	Dlinear	40.79		22771
	Lower bound	40.79		
2500/80/0.8/3	Greedy	21.98	7.9	685
	Greedy +	20.72	1.7	19179
	Dlinear	20.37		46545
	Lower bound	20.37		
2500/100/0.8/3	Greedy	17.14	5.2	740
	Greedy +	16.75	2.8	27452
	Dlinear	16.29		57906
	Lower bound	16.29		
2500/200/0.8/3	Greedy	8.56	5.1	1009
	Greedy +	8.37	2.8	12974
	Dlinear	8.15	0.1	116265
	Lower bound	8.14		
2500/500/0.8/3	Greedy	3.4	4.2	2313
	Greedy +	3.35	2.7	21430
	Dlinear	3.32	1.8	273048
	Lower bound	3.26		

Table 3 Experimental results on Zipf distributions, when $N = 2500$, $K = 50$, and $z = 3$

$N/K/\theta/z$	Algorithm	AED	% Error	Time
2500/50/0.5/3	Greedy	47.74	9.7	595
	Greedy +	46.02	5.7	23175
	Dlinear	43.52	0.02	29075
	Lower bound	43.51		
2500/50/0.7/3	Greedy	39.59	6.8	600
	Greedy +	38.47	3.8	23606
	Dlinear	37.05	0.02	29132
	Lower bound	37.04		
2500/50/0.8/3	Greedy	34.33	5.2	603
	Greedy +	33.49	2.6	24227
	Dlinear	32.61		29121
	Lower bound	32.61		
2500/50/1/3	Greedy	23.10	3.2	609
	Greedy +	22.53	0.6	27566
	Dlinear	22.38		28693
	Lower bound	22.38		

How long a tick lasts depends on several factors, such as the page size, the system available bandwidth, the broadcast technology, and the client devices. A tick is sufficiently long to allow both the server transmission and the client download.

By observing the tables, one notes that Greedy+ and Dlinear always outperform Greedy in terms of solution quality (that is, AED). In particular, Greedy + at least halves the error of Greedy, producing solutions whose errors is at most 5.7%. Moreover, Dlinear reaches the optimum almost in all cases, and its maximum error is as high as 1.8% only in one instance.

As regard to the running times, although all the three heuristics have the same $O(NK)$ time, Greedy is the fastest in practice. Indeed, its worst case instance is built ad hoc (see the Appendix) and never occurs in our experiments. Although Greedy+ and Dlinear are slower than Greedy, their running times are always less than one tenth of second. Their highest running times occur in Table 2, where those of Dlinear are directly proportional to K while those of Greedy+ are inversely proportional to K. This singular behavior of Greedy+ might depend on the fact that, in the second phase each Split execution stops its scan earlier when the cardinality of each pair of channels decreases, and therefore the number of channels K increases. It is worth to note that, due to the dynamic programming approach, Dlinear solves all the instances with $1 \leq n \leq N$ items and $1 \leq k \leq K$ channels, while Greedy and Greedy+ only solve the K instances with $n = N$.

The previous experiments have shown that Greedy+ and Dlinear behave well when the item popularities follow a

Table 4 Experimental results on a Zipf distribution, when $N = 500$, $K = 50$, and $\theta = 0.8$

$N/K/\theta/z$	Algorithm	AED	% Error	Time
500/50/0.8/3	Greedy	7.34	5.3	147
	Greedy +	7.19	3.1	2517
	Dlinear	6.98	0.1	5423
	Lower bound	6.97		
500/50/0.8/5	Greedy	10.78	5.3	147
	Greedy +	10.52	2.8	2938
	Dlinear	10.25	0.1	5490
	Lower bound	10.23		
500/50/0.8/7	Greedy	14.50	4.9	146
	Greedy +	14.16	2.4	3329
	Dlinear	13.85	0.2	5499
	Lower bound	13.82		
500/50/0.8/10	Greedy	19.48	5.1	145
	Greedy +	18.97	2.3	3899
	Dlinear	18.58	0.2	5507
	Lower bound	18.53		

Table 5 Experimental results on Stairs distributions, when $K = 20$, $z = 3$, $s = 6$, $b = 2$, and $\sigma = 0.8$

$N/K/z/s/b/\sigma$	Algorithm	AED	% Error	Time
500/20/3/6/2/0.8	Greedy	120165	7.8	101
	Greedy +	114005	2.3	1403
	Dlinear	111462	0.01	15590
	Lower bound	111437		
1500/20/3/6/2/0.8	Greedy	1062931	7.0	276
	Greedy +	1040415	4.8	28070
	Dlinear	992663	0.001	6282
	Lower bound	992647		
1750/20/3/6/2/0.8	Greedy	1449461	7.0	319
	Greedy +	1418920	4.7	38080
	Dlinear	1354459	0.005	7318
	Lower bound	1354384.66		
2000/20/3/6/2/0.8	Greedy	1888742	7.1	365
	Greedy +	1848449	4.8	50280
	Dlinear	1763132	0.0008	8392
	Lower bound	1763116.33		
2250/20/3/6/2/0.8	Greedy	2395045	7.2	410
	Greedy +	2343224	4.8	61358
	Dlinear	2234197	0.002	9773
	Lower bound	2234142.83		
2500/20/3/6/2/0.8	Greedy	2960463	7.4	459
	Greedy +	2897751	5.1	76332
	Dlinear	2756448	0.0009	10630
	Lower bound	2756421.5		

Zipf distribution. This suggests that, in most cases, the AED achieved in correspondence of the leftmost value of ℓ satisfying Recurrences 5 and 6 is the optimal AED or is very close to the optimal AED. In other words, the sequence of values obtained by varying ℓ is very often unimodal. In order to look for sequences that do not satisfy unimodality, and then to find critical instances for the above heuristics, a new distribution of popularities, called *Stairs*, is introduced. In such a distribution, there are few distinct, distant popularity values, with each value appearing many times. That is, the items to be broadcast are clustered by their popularities, with larger clusters having smaller popularities.

Formally, the Stairs distribution is determined by four parameters: the number N of items, the number s of distinct popularity values, the base value b, and the skewness σ. Specifically, the s distinct popularity values are $\{b, b^2, \ldots, b^s\}$, and there are Nq_{s+1-j} items with popularity b^j, where $q_1, \ldots, q_s$ are generated according to a Zipf distribution with skew σ. Note that the popularities can be normalized so that $0 \leq p_i \leq 1$ simply dividing each p_i by $\sum_{i=1}^{N} p_i = N \sum_{j=1}^{s} q_{s+1-j} b^j$. For instance, if $N = 12$, $s = 3$, $b = 2$, and $\sigma = 0$, one has $q_1 = q_2 = q_3 = \frac{1}{3}$, and

$$p_i = \begin{cases} 8 & \text{if } 1 \leq i \leq 4 \\ 4 & \text{if } 5 \leq i \leq 8 \\ 2 & \text{if } 9 \leq i \leq 12 \end{cases}$$

which can be normalized dividing by $12 \sum_{j=1}^{3} \frac{2^j}{3} = 56$.

Tables 5–7 report the results of the simulations for the Stairs distribution with the parameters $s = 4, 6$, $b = 2, 3$, and $\sigma = 0.8$, where the item popularities are not normalized, while the remaining parameters N, K, and z vary in the same ranges as before.

By observing the tables, one notes that both Greedy+ and Dlinear continue to outperform Greedy in terms of solutions quality. On the average, the errors of all heuristics are higher than those previously obtained for the Zipf distributions. However, Dlinear still continues to produce solutions very close to the optimum and its error is no larger than 0.8%.

For the sake of completeness, experimental tests are also performed on some uniform length benchmarks. In addition to the heuristics, also the Dichotomic algorithm is run in order to find the optimal solutions. In particular, Greedy+ is implemented by using the Split procedure, calling procedure BinSearch, shown in Fig. 3, while Greedy is implemented by using its original Split procedure shown in Fig. 5. Two sets of uniform length benchmarks are built, where the popularities are generated according to Zipf and Stairs distributions, respectively, N and K vary in the same ranges as for the non-uniform case, and the length z is fixed to 1. The results of the simulations are reported in Tables 8–11.

By observing the tables, one notes that Dlinear always finds the optimal solutions for Zipf distributions, while its maximum error is 1.6% for Stairs distributions. In both cases, Dlinear is about ten times faster than the optimal Dichotomic

Table 6 Experimental results on a Stairs distribution, when $N = 2500$, $z = 3$, $s = 4$, $b = 3$, and $\sigma = 0.8$

$N/K/z/s/b/\sigma$	Algorithm	AED	% Error	Time
2500/10/3/4/3/0.8	Greedy	9120504	9.1	380
	Greedy +	8522413.5	1.9	126305
	Dlinear	8358183	0.006	4800
	Lower bound	8358126		
2500/40/3/4/3/0.8	Greedy	2250825	8.3	599
	Greedy +	2136352.5	2.8	38527
	Dlinear	2076508.5	0.008	22313
	Lower bound	2076340		
2500/80/3/4/3/0.8	Greedy	1122418.5	8.2	678
	Greedy +	1073631	3.5	21415
	Dlinear	1037466	0.02	45853
	Lower bound	1037175.5		
2500/100/3/4/3/0.8	Greedy	914467.5	10.2	726
	Greedy +	869887.5	4.8	18946
	Dlinear	829735.5	0.004	58298
	Lower bound	829696.5		
2500/200/3/4/3/0.8	Greedy	454707	9.6	995
	Greedy +	429176	3.4	16034
	Dlinear	414792	0.01	114557
	Lower bound	414712		
2500/500/3/4/3/0.8	Greedy	170652	2.8	2332
	Greedy +	168804	1.7	18090
	Dlinear	167355	0.8	272210
	Lower bound	165892.62		

Table 7 Experimental results on a Stairs distribution, when $N = 500$, $K = 50$, $s = 6$, $b = 2$, and $\sigma = 0.8$

$N/K/z/s/b/\sigma$	Algorithm	AED	% Error	Time
500/50/3/6/2/0.8	Greedy	48943	9.9	141
	Greedy +	45543	2.3	2832
	Dlinear	44541	0.1	5347
	Lower bound	44495.66		
500/50/5/6/2/0.8	Greedy	71262	9.8	149
	Greedy +	66654	2.7	3159
	Dlinear	64973	0.1	5352
	Lower bound	64855.66		
500/50/8/6/2/0.8	Greedy	109899	9.8	142
	Greedy +	102227	2.1	4227
	Dlinear	100313	0.2	5423
	Lower bound	100065		
500/50/10/6/2/0.8	Greedy	129307	10.0	144
	Greedy +	119734	1.9	4540
	Dlinear	117755	0.2	5393
	Lower bound	117489.55		

Table 8 Experimental results on Zipf distributions, when $K = 20$, $\theta = 0.8$, and $z = 1$

$N/K/\theta/z$	Algorithm	AED	% Error	Time
500/20/0.8/1	Greedy	9.74	7.3	122
	Greedy +	9.17	1.1	177
	Dlinear	9.07		1948
	Dichotomic	9.07		9009
1500/20/0.8/1	Greedy	27.91	7.5	341
	Greedy +	26.70	2.8	228
	Dlinear	25.95		5863
	Dichotomic	25.95		30938
2000/20/0.8/1	Greedy	36.81	7.5	454
	Greedy +	35.20	2.8	263
	Dlinear	34.22		7890
	Dichotomic	34.22		42238
2500/20/0.8/1	Greedy	45.65	7.5	564
	Greedy +	43.62	2.8	279
	Dlinear	42.43		9909
	Dichotomic	42.43		55695

Table 9 Experimental results on a Zipf distribution, when $N = 2500$, $\theta = 0.8$, and $z = 1$

$N/K/\theta/z$	Algorithm	AED	% Error	Time
2500/10/0.8/1	Greedy	92.44	7.5	441
	Greedy +	86.85	1.0	170
	Dlinear	85.98		4576
	Dichotomic	85.98		26650
2500/40/0.8/1	Greedy	22.74	7.7	688
	Greedy +	21.88	3.6	442
	Dlinear	21.10		20689
	Dichotomic	21.10		116305
2500/80/0.8/1	Greedy	11.35	7.7	832
	Greedy +	10.79	2.4	670
	Dlinear	10.53		42238
	Dichotomic	10.53		238722
2500/100/0.8/1	Greedy	8.80	4.5	903
	Greedy +	8.63	2.4	796
	Dlinear	8.42		52982
	Dichotomic	8.42		298216
2500/200/0.8/1	Greedy	4.40	4.2	1243
	Greedy +	4.30	1.8	1390
	Dlinear	4.22		105940
	Dichotomic	4.22		602222
2500/500/0.8/1	Greedy	1.75	2.3	2624
	Greedy +	1.74	1.7	3299
	Dlinear	1.71		249843
	Dichotomic	1.71		1511243

algorithm. Due to the binary search used in the Split procedure, Greedy+ becomes the fastest heuristic and produces better sub-optimal solutions than Greedy.

In conclusion, although the difference in time to run the algorithms might seem long, all the algorithms are extremely fast in practice, since the slowest algorithm does not take more than one tenth of second. However, with such a modest increment in the running time, the slowest algorithm (i.e. Dlinear) provides a 5–10% better AED than the fastest one (i.e. Greedy). This means that in a realistic paging environment the client waits up to 10% ticks less.

Table 10 Experimental results on Stairs distributions, when $K = 20$, $z = 1$, $s = 4$, $b = 3$, and $\sigma = 0.8$

$N/K/z/s/b/\sigma$	Algorithm	AED	% Error	Time
500/20/1/4/3/0.8	Greedy	92115	6.2	117
	Greedy +	90879	4.8	139
	Dlinear	88065	1.6	1919
	Dichotomic	86658		8809
1500/20/1/4/3/0.8	Greedy	821439	6.2	338
	Greedy +	802851	3.8	189
	Dlinear	772788		5820
	Dichotomic	772788		30567
2000/20/1/4/3/0.8	Greedy	1455828	6.2	434
	Greedy +	1421523	3.7	206
	Dlinear	1370361		7779
	Dichotomic	1370361		41489
2500/20/1/4/3/0.8	Greedy	2277774	6.2	536
	Greedy +	223261	3.7	206
	Dlinear	2142918		9869
	Dichotomic	2142918		55124

```
Procedure Split $(i, j, m, \Delta)$;
    $m \leftarrow i$;
    $f \leftarrow C_{i,i} + C_{i+1,j}$;
    for $\ell$ from $i+1$ to $j-1$ do
        $temp \leftarrow C_{i,\ell} + C_{\ell+1,j}$;
        if $f \geq temp$ then
            $m \leftarrow \ell$ ;
            $f \leftarrow temp$
    $\Delta \leftarrow C_{i,j} - f$;
```

Fig. 5 The Split procedure used in the original Greedy heuristic

Table 11 Experimental results on a Stairs distribution, when $N = 2500$, $z = 1$, $s = 4$, $b = 3$, and $\sigma = 0.8$

$N/K/z/s/b/\sigma$	Algorithm	AED	% Error	Time
2500/10/1/4/3/0.8	Greedy	4622598	7.0	426
	Greedy +	4370205	1.2	156
	Dlinear	4316529		4509
	Dichotomic	4316529		25744
2500/40/1/4/3/0.8	Greedy	1137294	6.1	658
	Greedy +	1122519	4.7	325
	Dlinear	10780074	0.6	20689
	Dichotomic	1071630		115273
2500/80/1/4/3/0.8	Greedy	567351	5.8	802
	Greedy +	563565	5.1	585
	Dlinear	539913	0.7	42446
	Dichotomic	536019		236860
2500/100/1/4/3/0.8	Greedy	453681	5.7	855
	Greedy +	439260	2.4	633
	Dlinear	433611	1.1	53422
	Dichotomic	428850		297864
2500/200/1/4/3/0.8	Greedy	226908	5.7	1151
	Greedy +	221028	3.0	1090
	Dlinear	215415	0.4	105903
	Dichotomic	214497		601460
2500/500/1/4/3/0.8	Greedy	89070	3.4	2563
	Greedy +	88416	2.6	2935
	Dlinear	86436	0.3	252455
	Dichotomic	86127		1515853

5 Conclusions

In this paper, the problem of broadcasting data with non-uniform lengths over multiple channels, with the objective of minimizing the average expected delay of the clients, was considered under the assumptions of skewed allocation to multiple channels and flat scheduling per channel. Since for non-uniform lengths the problem is computationally intractable, new heuristics have been proposed, which experimentally outperform the previously known heuristic in terms of the solution quality. In particular, the experimental tests have shown that the Dlinear heuristic finds optimal solutions almost always. In contrast, Greedy is the fastest heuristic, but produces the worst solutions. Finally, Greedy+ presents running times and sub-optimal solutions which are both intermediate between those of Greedy and Dlinear. In conclusion, the choice among the heuristics depends on the goal to be pursued. If one is interested in finding the best sub-optimal solutions, then Dlinear should be adopted. Instead, if the running time is the main concern, then Greedy should be chosen, while if adaptability to parameter changes is the priority, then either Greedy or Greedy+ should be applied. In this scenario, Greedy+ represents a good compromise since it is scalable and produces fairly good solutions.

Appendix

This Appendix shows that the original Greedy algorithm presented in [17, 18] requires $O(NK)$ time in the worst case, instead of the claimed $O(N \log K)$ time, even for uniform length data items. However, it is shown that the $O(N \log K)$ bound holds in the average case.

The Greedy algorithm described in [17] is the same as the one given in Fig. 2 except for the Split procedure, which instead scans all the positions between i and j to find the best split point, as shown in Fig. 5.

In order to prove that the worst case time complexity of Greedy is $O(NK)$, consider N uniform length data items whose popularities are defined as follows:

$$p_i = \begin{cases} 1 & \text{if } N-3 \leq i \leq N \\ p_{i+1}(N-3) + \sum_{j=i+2}^{N} p_j + 1 & \text{if } 1 \leq i \leq N-4 \end{cases} \tag{7}$$

Lemma 5. *Consider $N - i + 1$ items, whose popularities $p_i, p_{i+1}, \ldots, p_N$ are generated by Eq. (7). Let $opt_{i,N,2}$ be the AED of an optimal solution S for assigning items $d_i, d_{i+1}, \ldots, d_N$ to two channels. Then $S = (i)$, that is d_i is assigned to one channel, and all the remaining items $d_{i+1}, \ldots, d_N$ are assigned to the other channel.*

Proof: By contradiction, assume that S is not an optimal solution. Then, consider the solution $S' = (i + 1)$ obtained from S by moving the border one position to the right. The cost of S' is given by $C_{i,i+1} + C_{i+2,N} = \sum_{j=i}^{i+1} p_j + \frac{(N-i-1)}{2} \sum_{j=i+2}^{N} p_j$. Since the cost of S is $C_{i,i} + C_{i+1,N} = \frac{1}{2}p_i + \frac{(N-i)}{2} \sum_{j=i+1}^{N} p_j$, S' is optimal if

$$\sum_{j=i}^{i+1} p_j + \frac{(N-i-1)}{2} \sum_{j=i+2}^{N} p_j < \frac{1}{2}p_i + \frac{(N-i)}{2} \sum_{j=i+1}^{N} p_j$$

This holds if and only if $p_i \leq p_{i+1}(N-3) + \sum_{j=i+2}^{N} p_j$, which contradicts Eq. (7). Thus, S' is not optimal. Moreover, by Lemma 4, further moving the border to the right can only increase the AED with respect to that of S'. Hence, $S = (i)$ is an optimal solution. □

Apply now the Greedy heuristic to the N data items $d_1, \ldots, d_N$ whose popularities are generated by Eq. (7) and to K channels, with $2 \leq K \leq N - 4$. By Lemma 5, every time Split is invoked, a new channel is added containing a single item. Precisely, the $K - 1$ Split invocations are: Split$(1, N)$, Split$(2, N), \ldots,$ Split$(K - 1, N)$. Hence, Greedy takes $\sum_{k=1}^{K-1} O(N - k) = O(KN)$ time.

To show that the $O(N \log K)$ bound holds in the average case, consider the kth invocation of Greedy. In this moment, the heap contains k elements, each specifying the indices i_r and j_r of the leftmost and rightmost item of the rth group, respectively. Clearly, such k elements correspond to a partition of the N data items into k segments, that is, sorting the elements by their rightmost indices, one obtains the segmentation $S = (j_{r_1}, \ldots, j_{r_{k-1}})$ built so far. Note that $i_{r_h} = j_{r_{h-1}} + 1$ for $2 \leq h \leq k$, with $i_{r_1} = 1$ and $j_{r_k} = N$. Once the rth element is extracted from the heap, the Split procedure performs $O(j_r - i_r)$ comparisons (see Fig. 5). Assuming that each element in the heap has the same probability $\frac{1}{k}$ to be extracted, the average time taken by the kth invocation of Greedy is

$$T(N, k) = O\left(\sum_{h=1}^{k} \frac{j_{r_h} - i_{r_h}}{k}\right) = O\left(\frac{N}{k}\right).$$

Therefore, the overall average time required by Greedy is

$$\sum_{k=1}^{K} T(N, k) = O\left(\sum_{k=1}^{K} \frac{N}{k}\right) = O(N \log K).$$

Acknowledgments The authors wish to thank W.G. Yee for having provided the code of the Greedy heuristic.

References

1. S. Acharya, R. Alonso, M. Franklin and S. Zdonik, Broadcast disks: data management for asymmetric communication environments, in: *Proc. SIGMOD* (May 1995).
2. M.H. Ammar and J.W. Wong, The design of teletext broadcast cycles, Performance Evaluation 5(4) (1985) 235–242.
3. M.H. Ammar and J.W. Wong, On the optimality of cyclic transmission in teletext systems, IEEE Transactions on Communications 35(11) (1987) 1159–1170.
4. E. Ardizzoni, A.A. Bertossi, M.C. Pinotti, S. Ramaprasad, R. Rizzi and M.V.S. Shashanka, Optimal skewed data allocation on multiple channels with flat broadcast per channel, IEEE Transactions on Computers 54(5) (2005) 558–572.
5. A. Bar-Noy, R. Bhatia, J.S. Naor and B. Schieber, Minimizing service and operation costs of periodic scheduling, in: *Proc. Ninth ACM-SIAM Symp. on Discrete Algorithms (SODA)* (1998).
6. L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, Web caching and Zipf-like distributions: evidence and implications, in: *Proc. IEEE INFOCOM* (1999).
7. S. Frattasi, R.L. Olsen, M. De Sanctis, F.H.P. Fitzek and R. Prasad, Heterogeneous services and architectures for next-generation wireless networks, in: *2nd IEEE International Symposium on Wireless Communication Systems* (2005) pp. 213–217.
8. A.S. Goldstein and E.M. Reingold, A Fibonacci version of Kraft's inequality applied to discrete unimodal search, SIAM Journal on Computing 22(4) (1993) 751–777.
9. T. Imielinski, S. Viswanathan and B.R. Badrinath, Energy efficient indexing on air, in: *Proc. SIGMOD* (May 1994).
10. C. Kenyon and N. Schabanel, The data broadcast problem with non-uniform transmission time, in: *Proc. Tenth ACM-SIAM Symp. on Discrete Algorithms (SODA)* (1999).
11. C. Kenyon, N. Schabanel and N. Young, Polynomial time approximation scheme for data broadcast, in: *Proc. ACM Symp. on Theory of Computing (STOC)* (2000) pp. 659–666.
12. S.-C. Lo and A.L.P. Chen, Optimal index and data allocation in multiple broadcast channels, in: *Proc. Sixteenth IEEE Int'l Conf. on Data Engineering (ICDE)* (February 2000).
13. W.C. Peng and M.S. Chen, Efficient channel allocation tree generation for data broadcasting in a mobile computing environment, Wireless Networks 9(2) (2003) 117–129.
14. K.A. Prabhakara, K.A. Hua and J. Oh, Multi-level multi-channel air cache designs for broadcasting in a mobile environment, in: *Proc Sixteenth IEEE Int'l Conf. on Data Engineering (ICDE)* (February 2000).
15. I. Stojmenovic (eds), *Handbook of Wireless Networks and Mobile Computing* (Wiley, Chichester, 2002).
16. N. Vaidya and S. Hameed, Log time algorithms for scheduling single and multiple channel data broadcast, in: *Proc. Third ACM-IEEE Conf. on Mobile Computing and Networking (MOBICOM)* (September 1997).

17. W.G. Yee, Efficient data allocation for broadcast disk arrays, Technical Report, GIT-CC-02-20, Georgia Institute of Technology (2001).
18. W.G. Yee, S. Navathe, E. Omiecinski and C. Jermaine, Efficient data allocation over multiple channels at broadcast servers, IEEE Transactions on Computers 51(10) (2002) 1231–1236.

Stefano Anticaglia received the bachelor's degree in Computer Science from the University of Perugia (Italy) in 2005. At present, he is a student in the master's of Computer Science of the University of Perugia.

Ferruccio Barsi received the doctor engineering degree from the University of Pisa, Italy, in 1969. From 1969 to 1992 he has been with he National Council of Research at the Istituto di Elaborazione dell'Informazione, Pisa. Since 1992, he is a Full Professor of Computer Science in the Mathematics and Computer Science Department of the University of Perugia, Italy.

His main contributions are in the areas of computer architecture, error-control coding, systems diagnosis, VLSI design, digital signal processing, and computer graphics. He is currently involved in researches concerning network security and wireless communications.

Alan Bertossi was born in London (England) in 1956. He got the Laurea Degree summa cum laude in Computer Science from the University of Pisa (Italy) in 1979. Afterwards, he worked as a System Programmer and Designer. From 1983 to 1994 he was with the University of Pisa as a Research Associate first, and later as an Associate Professor. From 1995 to 2002 he was with the University of Trento (Italy), as a Full Professor. Since 2002, he has been with the Department of Computer Science of the University of Bologna (Italy), as a Professor of Computer Science.

His main research interests are the computational aspects of high-performance, parallel, VLSI, distributed, fault-tolerant, and real-time systems. He has published about 40 refereed papers on international journals, as well as several papers in international conferences, workshops, and encyclopedias. He has authored a book (on design and analysis of algorithms, in Italian) and he served as a guest coeditor for special issues of Algorithmica, Discrete Applied Mathematics, and Mobile Networks and Applications. He is a member of the editorial board of Information Processing Letters. His biography is included in the 1999 edition of Who's Who in the World and in the 2000 edition of Who's Who in Science and Engineering. Since 1999, he has been a scientific collaborator at the Institute of Information Sciences and Technologies of the Italian National Research Council (ISTI-CNR, Pisa, Italy). During 2001–2003, he was the national coordinator of an Italian research project on algorithms for wireless networks.

Lucio Iamele received the bachelor's degree in Computer Science from the University of Perugia (Italy) in 2004. At present, he is working at Noranet (Italy) as a system programmer and designer.

M. Cristina Pinotti received the Dr. degree cum laude in Computer Science from the University of Pisa, Italy, in 1986. During 1987–1999 she was a Researcher with the National Council of Research at the Istituto di Elaborazione dell'Informazione, Pisa. From 2000–2003 she was an Associate Professor at the University of Trento. From 2004, she is a Full Professor at the University of Perugia. In 1994 and 1995 she was a Research Associate at the Department of Computers Sciences, University of North Texas, Denton, TX. In 1997 she visited the Department of Computer Science, Old Dominion University, Norfolk, VA (USA).

Her research interests are in wireless networks, sensor networks, design and analysis of algorithms, data broadcasting, channel assignment problems, graph coloring, multiprocessor interconnection networks, design and analysis of parallel algorithms, parallel data structures, distributed computer arithmetic, residue number systems, VLSI special purpose architectures. She has published about 50 refereed papers on international journals, in international conferences and workshops. She has been a guest co-editor for special issues of Mobile Networks and Applications, Wireless Networks and Journal of Parallel and Distributed Computing. She is a member of the editorial board of International Journal of Parallel, Emergent and Distributed Systems.