Broadcast Scheduling with Multiple Concurrent Costs

C. Liaskos*, Graduate Member, IEEE, A. Xeros†, G. Papadimitriou*, Senior Member, IEEE, M. Lestas‡ and A. Pitsillides†, Senior Member, IEEE

Abstract—Data dissemination via periodic broadcasting considers a set of items, each with a given request probability, size and scheduling cost. The goal is to construct a broadcast schedule that minimizes the mean query serving time and the mean scheduling cost at the same time. This task has been proven to be NP-Hard, and related studies have gradually discarded the scheduling cost attribute in an effort to simplify the problem. The present study reinstates the cost attribute, as well as any number of additional cost attributes per data item. The proposed, MULTIOPT scheduling algorithm then achieves optimal mean serving time and mean values for all costs concurrently. Comparison with brute-force results and related approaches yield optimality in all tested cases.

Index Terms—optimal periodic scheduling, multiple costs, wireless

I. INTRODUCTION

Ubiquitous data access is rapidly becoming the new trend in modern communications. Portable devices, such as smartphones and tablets enable human users and software agents to access the web anytime, at any place. The large degree of commonality in the accumulated data requests calls for direct exploitation through broadcasting. The resulting bandwidth savings are especially important in wireless environments, where the addition of extra physical paths is not an option. However, serving multiple queries with a single reply requires a degree of sophistication. Factors such as the mean serving time, user deadlines, broadcast copyright costs, energy efficiency at the client and at the server side must be considered concurrently. This fact highlights the need for proper broadcast scheduling, i.e. the serialization of data item broadcasts in a multiple criteria-optimal manner.

The evaluation of wireless broadcast scheduling techniques takes place in a widely approved system setup [1]–[16]. A number of wireless clients freely roam an area covered by a broadcast network. All clients read the broadcasted data stream synchronously, while wireless transmission parameters are idealized in order to focus on the evaluation of the scheduling process only. The data set to be broadcasted contains a number of discrete data items with known sizes. All item attributes may vary with time. Proposed scheduling techniques typically have an online expression [1]–[4], which can re-adapt the scheduling on the fly, based on new input regarding the client preferences or the update of the data set. The change in these parameters is detected and handled by smart learning algorithms which are examined elsewhere, as a separate field of study [5], [17]. Two types of scheduling are defined: in pull-based scheduling the clients pose specific queries to a server [18]. The server then serializes the answers to the requests in a way that minimizes the mean serving time or other criteria. In push-based (or periodic) scheduling, examined in the present study, the clients do not post queries to the server. A learning algorithm monitors the request probability of each available data item (or item class), typically through a lightweight, indirect feedback system [3] (e.g. by exploiting Facebook profile data in a subscription-based system). Thus the actual number of users is not directly relevant [4]. The goal is to utilize the item attributes (e.g. request probability and size) and produce a periodic schedule that optimizes the given criteria.

Existing periodic scheduling approaches do not support optimization of multiple criteria. This issue imposes practical limitations to the use of push-based broadcast systems. As an example consider the common case where each item has a copyright cost, which is effective per broadcast. A realistic scheduling authority would strive for a balance between the mean client serving time and the mean broadcasting cost. Additionally, assume that the physical server is a modern smartphone with hotspot functionality, and therefore battery lifetime is an issue. The scheduler should then balance the mean serving time, the mean cost and the mean expended energy at the same time. Nevertheless, current schedulers do not provide such functionality.

Originally, optimization of multiple criteria was an integral part of broadcast scheduling, albeit for just a single cost attribute per item [8]. The goal was the creation of a schedule that minimizes the mean client serving time and the mean scheduling cost at the same time. However, the problem was soon proven to be NP-Hard [6]. Furthermore, subsequent studies proved the NP-Hardness of several relaxations of the original problem [7]. Thus, the cost attribute was discarded, and related studies focused on the minimization of the mean serving time (or related metric) only [2]–[5], [11]–[16].

In the context of the present work we reinstate the broadcast cost attribute, and present a scheduler that can achieve minimal mean client serving time for any user-defined mean scheduling cost. The proposed, MULTIOPT scheduler is shown to be more efficient and tunable than the existing approaches. Additionally, an indefinite number of attributes can be added

* C. Liaskos and G. Papadimitriou are with the Department of Informatics, Aristotle University, 54124, Thessaloniki, Greece, e-mail: {cliaskos, gp}@csd.auth.gr.
† A. Xeros and A. Pitsillides are with the Department of Computer Science, Cyprus University, P.O. Box 20537 1678, Nicosia Cyprus, e-mail: {fcsp6xa1, lestas, andreas.pitsillides}@cs.ucy.ac.cy.
‡ M. Lestas is with the Department of Electrical Engineering at Frederick University, Nicosia, Cyprus. email: eng.lm@frederick.ac.cy.
per item, and the simultaneous minimization of their corresponding mean values is also feasible with near-perfect accuracy. MULTIOPT operates on the principles of alternating optimization [19], breaking down the multiple criteria optimization to an iterative series of single-criterion optimization. The cases of two and three concurrent criteria are examined through extensive simulations and are shown to coincide with brute-force results. It is clarified that the proven theoretical NP-Hardness of the problem is not alleviated, but rather shown not to be prohibitive in practical communication systems.

II. RELATED WORK

Research on push-based broadcast systems initially focused on the minimization of the clients’ mean serving time, over an infinite time horizon, in the context of Teletext systems [9]. It was proven that an optimal schedule comprises periodic occurrences of each data item. Therefore, one needs to define only the optimal number of occurrences of each data item inside the schedule. Authors in [9] provided an analytical solution, assuming equally-sized data items. The problem was revisited in [10], heuristically studying items with small variation in their sizes. It was clarified that the mean serving time depends on data item attributes (i.e. item request probabilities and sizes), and not on the number of clients. The square root rule for optimal scheduling and the lower bound for the mean serving time were formally defined in [4]. The same study proposed practical scheduling algorithms that achieved optimality. However, the schedule size infinity assumption resulted in increased complexity. Authors in [2] presented an analysis-derived periodic scheduler that achieved the same performance with minimal schedules, for any item sizes. The broadcasting process was divided into a scheduling and a serializing stage. The scheduling stage defines the optimal number of periodic item occurrences in the schedule, which yield minimal waiting time. The serializing process is generic, receiving any item occurrences and producing the corresponding series of actual, periodic item broadcasts. The low complexity of [2] pertains to both stages as a whole.

Heuristic, low complexity scheduling methods were introduced in [1] with the introduction of the Broadcast Disks model. According to it, items are grouped by popularity, forming virtual disks rotating around a common axis. Imaginary heads read and serialize data from the disks, producing the final schedule. In [11] the authors applied clustering techniques for performing the data grouping. In [12] the grouping of items was analytically optimized. The analytical results were exploited in [13] for producing minimal complexity schedulers. All these studies focused on the minimization of the clients’ mean serving time.

Further studies have addressed the issues of multi-channel broadcasting [15], serving correlated items [16], pull-push hybrid data dissemination [20]–[22] and data indexing for saving power at the client devices. The study of [23] proposed a smart data packet format that combines double-linked list traversing and peek forward/backward capabilities. A client device reads the packet header info and schedules the activation of the wireless module correspondingly, thus conserving energy. The concept is extended in [24] via the Windmill Scheduling Algorithm, which combines optimal periodic scheduling with the energy-conserving data indexing. Concerning the application context, multimedia streaming has been extensively studied [25], [26], especially in DVB-T/H scenarios [27], [28].

As previously stated, a more strict version of the scheduling problem assigns an additional attribute, the scheduling cost, to each data item. The new goal is to define the schedule that minimizes the mean serving time and the mean scheduling cost at the same time. The authors in [7] proved that the corresponding optimal schedule is also periodic in this case as well. The problem is then mapped to the generalized maintenance scheduling problem, which is a known NP-Hard problem. Several greedy algorithms are presented, as well as in [6], which generally operate beyond the analytically optimal bounds. To the best of the authors knowledge, the study of [7] constitutes the prevailing solution to the problem in the context of the studied push-based systems. Furthermore, the issue of optimizing an indefinite number of criteria through a single schedule has not been previously addressed.

The remainder of this paper is organized as follows: Section III presents the analysis leading to the definition of the novel, MULTIOPT scheduler. Comparison with related work is provided in Section IV-A. Multiple cost attributes per item are compared to brute-force-derived results in Section IV-B. Conclusion is given in Section V.

III. MATHEMATICAL ANALYSIS

A. Standard assumptions and notation

We regard a set of $N$ data items arbitrarily indexed by $i = 1 \ldots N$. Each item $i$ is associated with its size $l_i$ (in bytes), its request probability $p_i$, $\sum_{i=1}^{N} p_i = 1$ and a set of scheduling costs $c_j$, $j = 1 \ldots M$ which are normalized in $[0, 1]$. Furthermore, $u_i \in \mathbb{N}^N$ denotes the number of periodic occurrences of item $i$ in the schedule. The total size of the schedule is:

$$L = \sum_{i=1}^{N} u_i \cdot l_i$$

while the mean $j^{th}$ cost is:

$$C_j = \frac{\sum_{i=1}^{N} u_i \cdot c_{ij}}{\sum_{i=1}^{N} u_i}, \quad j = 1 \ldots M$$

No assumptions are made concerning the nature of a data item during the analysis. In accordance with the related work on scheduling, an item is simply a piece of information that a client may acquire through a single query [1]–[4], [6], [7], [9]–[16], [29], [30]. It is clarified that in push-based, periodic broadcast scheduling, the term “client query” does not imply posting a request to a server, but rather waiting for the broadcast of a specific item. In a periodic schedule, the quantity $l_i/u_i$ represents the interval between two consecutive occurrences of item $i$. Therefore, the mean waiting time is:

$$\bar{W} = \sum_{i=1}^{N} p_i \cdot \frac{L}{2 \cdot u_i} = \frac{1}{2} \left( \sum_{i=1}^{N} p_i \cdot l_i \right) \cdot \left( \sum_{i=1}^{N} \frac{p_i}{u_i} \right)$$

Notice that $\bar{W}$ does not depend on the number of clients, which are handled collectively as a Gaussian process via the
central limit theorem [4]. In addition, (3) measures \( W \) in size units (e.g. bytes). Conversion to time units requires the definition of a physical wireless transmission rate. Equation (3) is minimized when [4]:

\[
u_i \propto \sqrt{p_i/l_i}
\]  

(4)

which is known as the square root rule.

B. The case of a single cost revisited

We consider the case of a single cost, \( c_{i1} \), per item, i.e. \( M = 1 \). This attribute corresponds to the scheduling cost of [6], [7], and is open to any physical interpretation that can be efficiently expressed in the value set \( (0,1) \). The goal is to define the item occurrences, \( u_i \), that minimize the mean cost \( \overline{C_1} \) and the mean waiting time \( \overline{W} \).

Notice that the mean cost \( \overline{C_1} \) is minimized when \( u_i = 0, \forall i \notin \text{argmin}_{(1)} \{c_{i1}\} \). The notation \( \text{argmin}_{(1)} \{ \ast \} \) denotes the lowest value of \( (\ast) \) for which the condition \( \{ \ast \} \) holds. In other words, the mean cost is minimized when we exclusively broadcast the item with the lowest cost. On the other hand, \( \overline{W} \) is minimized when the relation (4) holds. Furthermore, the cost attributes, \( c_{i1} \), are not correlated in any way to the remaining item attributes, \( p_i, l_i \). Consequently, the literal expression “concurrent minimization of \( \overline{C_1} \) and \( \overline{W} \)” has initially no physical meaning. Minimizing \( \overline{C_1} \) and \( \overline{W} \) concurrently requires the definition of a metric that combines both quantities. For example, [6] and [7] assume that both \( \overline{C_1} \) and \( \overline{W} \) are of equal importance and define the combination:

\[
S = 50\% \cdot \overline{C_1} + 50\% \cdot \overline{W}
\]  

(5)

which needs to be minimized. However, the equal importance assumption is restrictive. In order to overcome this shortcoming, the following analysis will derive the full relation \( \overline{W} = f(\overline{C_1}) \), i.e. the schedule that minimizes \( \overline{W} \) for any given cost \( \overline{C_1} \). Any custom metric can then be satisfied at the intersection of the plot \( \overline{W} = f(\overline{C_1}) \) and the line \( \overline{W} = b \cdot \overline{C_1}, b > 0 \). As an example, the combination \( S \) of equation (5) represents the very specific case of intersecting \( \overline{W} = f(\overline{C_1}) \) and \( \overline{W} = \overline{C_1} \).

In order to enable the use of infinitesimal calculus, we will expand the value set of \( u_i \) from \( \mathbb{N} \) to \( \mathbb{R}^+ \). Indeed, if \( u_i \) is very large \( \forall i \), one can safely assume that \( u_i \pm 0.5 \approx u_i \), where \( \pm 0.5 \) represents any rounding error. Equation (2) then relates \( u_i \) to any arbitrary \( u_k \), \( k = 1 \ldots N, k \neq i \) as follows:

\[
\frac{\partial u_k}{\partial u_i} = \frac{-C_{i1} - c_{i1}}{C_{i1} - c_{k1}}
\]  

(6)

Taking the first derivative of the mean client serving time, \( \overline{W} \) - equation (3), with regard to \( u_i \), produces through (6):

\[
\frac{\partial \overline{W}}{\partial u_i} = \frac{1}{2} \left[ l_i - l_k \frac{C_{i1} - c_{i1}}{C_{i1} - c_{k1}} \right] A_1 \cdot \left( \sum_{m=1}^{N} \frac{p_m}{u_m} \right) B_1 + \ldots
\]

\[
+ \frac{1}{2} \left[ -\frac{p_i}{u_i^2} + \frac{p_k}{u_k^2} \frac{C_{i1} - c_{i1}}{C_{i1} - c_{k1}} \right] A_2 \cdot \left( \sum_{m=1}^{N} u_m \cdot l_m \right) B_2
\]

(7)

The reader may notice that the described procedure is an application of the Lagrange method of restricted optimization, on equations (2) and (3). The labels \( A_{1,2}, B_{1,2} \) are added for quick referencing. Concerning the possible nullification of the \( (\overline{C_1} - c_{k1}) \) denominator, we simply note that \( c_{k1} \) (i.e. reference item \( k \)) can be chosen to be different from the user-defined mean cost \( \overline{C_1} \), which is supplied as an input. Minimizing the mean serving time entails the nullification of equation (7) for every \( i \) and \( k \). In order to facilitate the solution to the problem, we make the following remark:

**Remark 1.** Assume a solution, \( u_{i^*} \), \( i = 1 \ldots N \), which nullifies equation (7). Any set \( u^*_i = \lambda \cdot u_{i^*} \), \( \lambda > 0 \) is also a valid solution.

**Proof:** We insert \( u_{i^*} = \lambda \cdot u_i \) in equation (7), which leads to:

\[
\frac{\partial \overline{W}}{\partial u_i} = \frac{1}{2} \lambda \cdot \frac{\partial \overline{W}}{\partial u_i} = 0
\]

(8)

The restriction \( \lambda > 0 \) ensures that the new solution, \( u^*_i \), comprises only positive values, successfully representing item occurrences. This concludes the proof.

The preceding remark states that the optimal solution to the scheduling problem in question can be expressed as a ratio of item occurrences. Therefore, the occurrences \( u_k \) of reference item \( k \) in equation (7) can be set in any convenient way. For uniformity and simplification reasons, we define that:

\[
u_k = \sqrt{p_k/l_k}
\]  

(9)

The complete solution is then given through the following Theorem.

**Theorem 2.** The optimal, periodic item occurrences, \( u_i \), that correspond to a given mean cost, \( \overline{C_1} \), and minimize the mean client waiting time, \( \overline{W} \), are:

\[
u_i = \sqrt{\frac{l_i - l_k \frac{C_{i1} - c_{i1}}{C_{i1} - c_{k1}} \cdot V + l_k \cdot \frac{C_{i1} - c_{i1}}{C_{i1} - c_{k1}}}{p_i}} , \ i = 1 \ldots N
\]

(10)

where \( V > 0 \) is calculated numerically through:

\[
\overline{C_1} = \sum_{i=1}^{N} u_i \cdot c_{i1}/\sum_{i=1}^{N} u_i
\]

(11)

and reference item \( k \) is selected as:

\[
k = \text{argmin}_{(k)} \left\{ \left( \sum_{i=1}^{N} p_i \cdot l_i \right) \cdot \left( \sum_{i=1}^{N} \frac{p_i}{u_i} \right) \right\}
\]

(12)

**Proof:** Quantities \( B_1 \) and \( B_2 \) of (7) are inherently positive. Therefore, in order for the \( \partial \overline{W}/\partial u_i \) derivative to be nullified, it must either hold that i) \( A_1 = A_2 = 0 \) or ii) \( A_1, A_2 \neq 0 \).
The first case holds only when \( i = k, \) i.e., when setting the number of occurrences of the reference item. Consequently, for each \( i \neq k \) it must hold that \( A_1, A_2 \neq 0. \) Inserting equation (9) in (7) and solving for \( u_i \) yields:

\[
u_i = \sqrt{\frac{p_i}{l_i - l_k \cdot \frac{\overline{C}_{i} - c_{i1}}{\overline{C}_{i1} - c_{i1}}} \cdot \frac{B_2}{B_2 + l_k \cdot \frac{\overline{C}_{i} - c_{i1}}{\overline{C}_{i1} - c_{i1}}}}
\] (13)

The quantities \( B_1 \) and \( B_2 \) of equation (7) constitute reductions of the complete solution set, \( u_i, i = 1 \ldots N. \) Therefore it holds that \( V = B_1/B_2 = \text{constant}, \) for every \( i. \) Thus, equation (10) is formed. In order to define the value of \( V, \) we employ equation (2) for the requested \( \overline{C}_1 \) value. The procedure has so far defined the optimal item occurrences, given a selected reference \( k. \) Thus, in (12), we select the reference item that corresponds to minimal mean serving time (equation (3)) over all other choices.

A consequence of Theorem 2 is that certain requested mean cost values \( \overline{C}_1 \) are not feasible. The denominator of equation (10) must be positive due to the square root. However, there is no guarantee that there will always exist a positive \( V \) value for which (11) holds. This fact is summarized in the following Corollary.

**Corollary 3.** A requested mean cost \( \overline{C}_1 \) can be achieved only if equation (11) has a solution for \( V \) in the range defined by:

\[
[l_i - l_k \cdot \frac{\overline{C}_1 - c_{i1}}{\overline{C}_{i1} - c_{i1}}] \cdot V > -l_k \cdot \frac{\overline{C}_1 - c_{i1}}{\overline{C}_{i1} - c_{i1}}, \quad i = 1 \ldots N
\] (14)

If not, there does not exist a schedule that achieves minimal mean waiting time for the given cost.

A simple example pertaining to Corollary 3 is that a mean scheduling cost below \( \min\{c_{i1}\} \) or above \( \max\{c_{i1}\} \) is not achievable by any real schedule. A procedural format of the described process is given via the OPT-1 Algorithm.

The OPT-1 algorithm tries out all items as potentially optimal references, \( k \) (for-loop at step 3), as dictated by equation (12). A core procedure (steps 4 – 7) is repeated for each option, eventually keeping the one that yields the smallest mean waiting time (steps 8 – 11). The core procedure consists of solving equation (11) numerically for \( V, \) in the range defined by (14). The present study uses the Levenberg-Marquardt numerical method [31]. Upon finding a valid solution, the optimal item occurrences are calculated. If there exists no solution for \( V, \) \( NULL \) values are returned, indicating that the chosen mean cost value, \( \overline{C}_1, \) is impossible to achieve. The algorithm returns the denominator of equation (10) as \( l_1^* \) (step 14), which will prove useful when examining multiple concurrent costs. Finally, the algorithm has \( O(N \cdot B) \) complexity, \( B \) being the complexity of the chosen numerical method.

**C. Extension to additional cost attributes per item**

Theorem 2 states that even when one needs to optimize the mean waiting time for a given mean cost, a square root rule similar to (4) governs the optimal item occurrences. Notice that (4) refers to the minimization of the mean waiting time only and discards the cost attributes altogether. However, the similarities are readily noticeable. Both solutions take the form of ratios (Remark 1), which are expressed as the square root of the request probability of an item divided by a quantity measured in size units. This observation is more apparent when equation (10) is rewritten as:

\[
u_i = \sqrt{\frac{p_i}{l_i}}, \quad i = 1 \ldots N
\] (15)

where

\[
l_i^* = \left[l_i - l_k \cdot \frac{\overline{C}_1 - c_{i1}}{\overline{C}_{i1} - c_{i1}}\right] \cdot V + l_k \cdot \frac{\overline{C}_1 - c_{i1}}{\overline{C}_{i1} - c_{i1}}
\] (16)

In essence, the presence of a cost attribute \( c_{i1} \) forces the system to perceive the physical size of the items differently. From another point of view, the problem of minimizing the mean waiting time for a given mean cost can be substituted by an equivalent problem of sole mean waiting time minimization. After transformation (16) has been applied, the problem can be solved through the classic square root rule (15).

Assume that there exist \( M \) attributes, \( c_{ij}, j = 1 \ldots M, \) for each data item, \( i. \) We proceed to apply transformation (16) \( M \) consecutive times. The \( n^{th} \) application is expressed as:

\[
l_i^{(n)} = \left[l_i^{(n-1)} - l_k^{(n-1)} \cdot \frac{\overline{C}_1 - c_{i,n}}{\overline{C}_n - c_{(n),n}}\right] \cdot V(n) + \ldots
\]

\[
\ldots + l_k^{(n-1)} \cdot \frac{\overline{C}_1 - c_{i,n}}{\overline{C}_n - c_{(n),n}}
\] (17)

The expressions \( V(n), k(n) \) simply denote dependance on the current step since they must be calculated through (11) and (12) prior to completion. Furthermore, \( \overline{C}_n \) refers to the \( n^{th} \) of the concurrently requested cost values, \( \{\overline{C}_1, \overline{C}_2, \ldots \}

**Algorithm 1 Single-criterion OPTimization (OPT-1)**

**INPUT:**
- A set of \( N \) items, \( \{p_i, l_i, c_{i1}\}, i = 1 \ldots N. \)
- The desired mean cost \( \overline{C}_1. \)

**OUTPUT:**
- The optimal item occurrences \( u_i^*, i = 1 \ldots N \) that achieve mean cost \( \overline{C}_1 \) and minimal waiting time.
- The denominator of equation (10), \( l_1^*. \)

1: Initialize \( u_i^* = NULL, i = 1 \ldots N; \)
2: Set \( \min WT = Inf; \)
3: FOR \( k = 1 \ldots N, \)
4: Numerically solve eq. (11) for \( V \) of eq. (10), under restrictions (14);
5: IF \( V \) exists
6: Set \( u_i \) via eq. (10);
7: Calculate \( \overline{W} \) from eq. (3);
8: IF \( \overline{W} < \min WT \)
9: Set \( \min WT = \overline{W}; \)
10: Set \( u_i^* = u_i, i = 1 \ldots N; \)
11: ENDIF
12: ENDIF
13: ENDFOR
14: Set \( l_1^* = p_i/(u_i^*)^2, i = 1 \ldots N; \)
It produces the items \( \{7\} \).

Upon completion of the \( n \)th application, equation (15) will yield a solution \( \{u_i^{(n)} \}_{i=1}^N \) that indeed achieves a mean cost \( \overline{C}_n \), as well as an optimal corresponding mean waiting time (since \( \partial W/\partial u_i \) is nullified). However, an issue arises at the \((n + 1)\)
th application. The new solution, \( \{u_i^{(n+1)} \}_{i=1}^N \), will yield the requested cost \( \overline{C}_{n+1} \), but will upset the \( \overline{C}_n \) value achieved at the previous step. A way is needed for locking the produced solutions to the requested operation point \( \{\overline{C}_1, \overline{C}_2, \ldots, \overline{C}_M\} \). Notice that the \( n \)th step yields absolute convergence to \( \overline{C}_n \) and limited divergence from \( \overline{C}_j \), \( j \neq n \). Therefore, repeatedly applying all \( M \) steps of equation (17) could serve as the needed lock, eventually converging to the requested operation point. This approach is formulated as the \textsc{MULTIOPT} algorithm.

**Algorithm 2 MULTIple-criteria OPTimization (MULTIOPT)**

**INPUT:**
- A set of \( N \) items, \( \{p_i, l_i, \{c_{ij}, j = 1 \ldots M\} \}_{i=1}^N \).
- The desired mean costs \( \overline{C}_d = \{\overline{C}_1, \ldots, \overline{C}_M\} \).
- An acceptable error tolerance \( \varepsilon \in \mathbb{R}^+ \) for \( \overline{C}_d \).
- Maximum allowed iterations, \( X \in \mathbb{N}^* \), for achieving \( \varepsilon \).

**OUTPUT:** The item occurrences \( u_i, i = 1 \ldots N \) that achieve the mean costs and minimal waiting time.

1. Initialize a \( 1 \times M \) null vector, \( \overline{C}_r \).
2. WHILE \( \left| \overline{C}_d - \overline{C}_r \right| > \varepsilon \) AND \( X > 0 \)
3. \( X = X - 1 \);
4. FOR \( j = 1 \ldots M \)
5. Set \( \{l_i\} = \ldots OPT - 1 \{p_i\}, \{l_i\}, \{c_{ij}, i = 1 \ldots N\}, \overline{C}_d[j] \)
6. ENDFOR
7. Set \( u_{ij} = \sqrt{p_i/l_i}, i = 1 \ldots N \)
8. Set \( \overline{C}_r[j] = \sum_{i=1}^N u_{ij}/\sum_{i=1}^N u_{ij}, j = 1 \ldots M \)
9. END

The algorithm receives \( N \) data items as input, each one having \( M \) associated cost attributes, \( c_{ij} \). It produces the item occurrences \( u_i \) which achieve \( M \) user defined mean cost values supplied in vector form, \( \overline{C}_d = \{\overline{C}_1, \ldots, \overline{C}_M\} \), as well as minimal waiting time. Notice that the requested costs may not be achievable due to restrictions similar to (14). In order to detect this outcome, a user-specified precision, \( \varepsilon > 0 \), and a maximum number of iterations, \( X \), are supplied. Typical values are \( 10^{-M} \) and \( X = 100 \). If the specified precision has not been achieved after \( X \) iterations, the costs \( \overline{C}_d \) are not achievable. The precision refers to the \( \left| \overline{C}_d - \overline{C}_r \right| \) quantity, where \( \overline{C}_r \) are the projected costs at the current iteration, calculated at step 9 via equation (2). Finally, the core of each iteration consists of applying the \( OPT - 1 \) algorithm consequitively, each time using the \( l_i^* \) output of the previous application as the current input (lines 4 – 6). The overall complexity is \( O(M \cdot N \cdot B) \), where \( B \) is the complexity of the numerical method employed in \( OPT - 1 \).

\textsc{MULTIOPT} can be classified as an alternating optimization algorithm. Alternating optimization targets the optimization of a function \( f(\vec{x}) \), \( \vec{x} \) being a vector of indefinite size whose optimal value must be defined. Instead of attempting the optimization directly, alternating optimization targets a single attribute of the \( \vec{x} \) at a time, e.g. \( x_1 \). Optimizing \( x_1 \) alters the values of the remaining attributes \( x_2 \ldots \). We then proceed to optimize \( x_2, x_3, \ldots \) and repeat the process iteratively in a round robin manner until convergence has been achieved. \textsc{MULTIOPT} operates on this logic, considering the multiple costs as the vector \( \vec{x} \) of alternating attributes.

**IV. SIMULATION RESULTS**

This section compares the \textsc{MULTIOPT} algorithm to bruteforce-derived results as well as to related approaches [7]. Notice that the ability to handle multiple costs concurrently has not been addressed prior to the present study. Therefore, for two or more costs, the comparison takes places between \textsc{MULTIOPT} and bruteforce only. In the case of a single cost, \textsc{MULTIOPT} is also compared to the \textsc{Greedy}, \textsc{Periodic} and \textsc{Randomized} schedulers proposed in [6], [7]. As discussed in Section III, the later refer to the optimization of criterion (5). The single and multiple-cost cases are presented in the corresponding subsections that follow.

Bruteforce enables comparison with truly optimal values, but poses limitations as well. The procedure consists of checking all possible combinations of item occurrences \( u_i = 1, 2, \ldots, 500 \), for all \( i = 1 \ldots N \). A full simulation is run for each combination, which is also repeated 100 times in order to diminish the variance of the results to satisfactory levels. Therefore, keeping the runtimes feasible imposed a maximum of \( N = 5 \) items and \( M = 3 \) concurrent costs. Furthermore, the \( \max\{u_i\} = 500 \) restriction means that some extreme cases may not be satisfactorily approachable for the bruteforce procedure (e.g. the case of solely broadcasting one item). However, these cases are of limited practical interest and represent a very small fraction of the whole solution set.

The evaluation is based on the simulation environment of [2], [11], [13], implemented in \textsc{MATLAB™} [32]. A star topology comprises a central scheduling and broadcasting server and a number of 1,000 wireless clients. The number is purposefully large enough for the central limit theorem to apply on the totality of the client’s queries safely [4]. All clients are within the range of the server and wireless propagation and noise issues are idealized in order to focus on the evaluation of the proposed scheduler [1], [2], [4], [6], [7], [11], [13]. By the same convention, time is measured in \textit{Bytes}, and conversion to \textit{seconds} requires the definition of a data transfer rate. Since physical layer issues are idealized, the reader is encouraged to adopt any data transfer rate that corresponds to a physical system of interest (e.g. DTV in [33] or collaborative video streaming in [34]).

We consider data items with sizes \( l_i \in [10^{-3}, 1] \text{ MBytes} \), which correspond to typical, web-style data sites containing text, images, sound and animation. The items’ request probabilities \( p_i \)s are set according to the Zipf distribution [35], as in the majority of the cited studies. The parameter \( \theta \) is an input, denoting the skewness parameter of the distribution.
(\(p_i \propto 1/\theta\)). The present graphs in this section correspond to \(\theta = 0.95\) by convention. Other tested values, \(\theta \in [0, 1.5]\), produced similar results.

Each item \(i\) is associated with three costs, \(c_{i1}, c_{i2}, c_{i3}\). We assume that \(c_{i1}\) represents expended energy per broadcast of item \(i\). Each item requires energy due to the use of the server’s wireless module. However, a given item may require GPGPU pre-processing, retrieval from a remote database or direct hard disk access, adding up to the required energy per broadcast. Other items may be static and cached in RAM, requiring much less energy. The normalized \(c_{i1}\) attributed expresses this difference in power requirements. In a similar fashion, \(c_{i2}\) expresses information exclusiveness per item \(i\). An item may become available sooner through other, external sources. Thus, the \(c_{i2}\) attribute expresses the ratio of queries for item \(i\) that are overtaken by other systems. Finally, \(c_{i3}\) expresses copyright costs per broadcast of item \(i\). The server seeks to balance the expended energy rate, the query “leak” rate, the copyright costs and the mean client waiting time. This is accomplished by defining appropriate mean values, \(\{T_1, T_2, T_3\}\), pertaining to the \(c_{i1}, c_{i2}, c_{i3}\) attributes. The employed attributes values are given in Table I and are normalized in \([0, 1]\) for presentational purposes. Multiplication with any reference quantity can map them to context-specific values.

<table>
<thead>
<tr>
<th>attribute</th>
<th>(c_{i1})</th>
<th>(c_{i2})</th>
<th>(c_{i3})</th>
<th>(p_i)</th>
<th>(l_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_{i1})</td>
<td>0.6324</td>
<td>0.7975</td>
<td>0.2785</td>
<td>0.5499</td>
<td>0.2945</td>
</tr>
<tr>
<td>(c_{i2})</td>
<td>0.4218</td>
<td>0.9157</td>
<td>0.7292</td>
<td>0.5956</td>
<td>0.8296</td>
</tr>
<tr>
<td>(c_{i3})</td>
<td>0.6507</td>
<td>0.3957</td>
<td>0.6891</td>
<td>0.9340</td>
<td>0.5567</td>
</tr>
<tr>
<td>(p_i)</td>
<td>0.4787</td>
<td>0.2422</td>
<td>0.1647</td>
<td>0.0653</td>
<td>0.0600</td>
</tr>
<tr>
<td>(l_i)</td>
<td>0.8147</td>
<td>0.9058</td>
<td>0.1270</td>
<td>0.9134</td>
<td>0.1384</td>
</tr>
</tbody>
</table>

Table I: Simulation configuration - Item attribute values

Every simulation run is initialized as follows: A number of 300,000 item queries are generated according to the ZIPF distribution with the given \(\theta\) value. These are assigned to 1,000 clients (300 items per client). A client may have one request pending at any given time (single-threaded clients). This incurs no loss of generality since a client with \(m\) simultaneous queries can be substituted by \(m\) single-threaded clients. For a given set of required mean values \(\{T_1, T_2, T_3\}\), the MULTIOPT, Greedy, Periodic and Randomized algorithms define the optimal number of item occurrences. The generic periodic serializer of [2] then constructs the broadcast schedule and transmits it periodically to the clients. If a broadcasted item \(i\) coincides with the current query of a client, the query is discarded and the client moves to the next query after a random \(\text{ThinkTime}\) of \([0, 3]\) \(\text{MBytes}\). The latter serves the purpose of taking into account the time a human user requires to process the data before defining the next query. The simulation ends with the depletion of all client queries.

A. Single cost attribute per item

Figure 1 corresponds to a single active cost per item. Specifically, it illustrates the case of balancing the energy consumption rate \(c_{i1}\) attributes) and the achieved mean waiting time of the clients. The proposed MULTIOPT algorithm produces identical results with the brute force procedure. Notice that mean energy consumption greater than \(\text{max}\{c_{i1}\}\) or smaller than \(\text{min}\{c_{i1}\}\) are not achievable as expected. The MULTIOPT and the brute force results coincide in this aspect as well. The Greedy, Periodic and Randomized schedulers can operate for a single \(T_1\) value only, offering no tunability. For this operational value, they tend to behave near-optimally achieving mean waiting times comparable to the bruteforce process. The Randomized scheduler is the sole exception to this trend. This scheduler employs random intervals between item occurrences in the schedule. However, periodicity has been proven to be a prerequisite for minimal waiting time [9]. Therefore, the Randomized scheduler behaves sub-optimally. More importantly, the Greedy, Periodic and Randomized schedulers tend to pursue mainly the minimization of the mean waiting time. Notice that the corresponding operation points in Fig. 1 revolve around the global minimum. Other studies, that target the sole minimization of the mean waiting time without active costs, achieve similar performance [2], [4]. This fact raises fitness issues for custom optimization metrics, such as equation (5) which is employed by the compared schedulers.

![Simulation results on the achieved mean client waiting time versus the required energy consumption rate at the server. MULTIOPT offers maximum tunability in terms of selecting the energy consumption rate and optimal performance in any case. The Greedy, Periodic and Randomized schedulers can operate only at single point, which cannot be altered by the scheduler.](image1)

On the other hand, the MULTIOPT algorithm imbues the scheduling process with tunability since it is able to achieve optimal performance for any user-specified \(T_1\) value.

Finally, the cases of query leak rate \(c_{i2}\), copyright costs \(c_{i3}\) Vs the mean waiting time produce results similar to those of Fig. 1.

B. Multiple cost attributes per item

Figure 2 examines the case of two concurrent costs per item \(i\). The server strives to create a schedule that yields an acceptable, pre-specified energy consumption rate \(c_{i1}\) and a specific mean copyright cost \(c_{i3}\), while minimizing the client waiting time. The illustration requires a 3D plot of the mean waiting time on the plane \([T_1 \times T_3]\). Illustration purposes impose the use of contour plots in Fig. 2a. It is noted that the bruteforce results and the MULTIOPT performance coincide to the point of not exhibiting visual difference. This fact qualitatively indicates optimal performance for the MULTIOPT
algorithm. In order to study any differences quantitatively, a diversion distribution graph is given in Fig 2b. The diversion graph comprises a histogram of the signed difference,

$$\Delta = WaitingTime_{MULTIOPT} - WaitingTime_{BruteForce}$$

(18)

Notice that all $\Delta$ values in Fig. 2b are negative or zero. Specifically, the bulk of the $\Delta$ values (95%) are null, while a small percentage (5%) are negative. This fact indicates that the MULTIOPT algorithm produces equally good or better results than the brute force procedure. Attributing the negative results to the limitations of the brute force process described earlier, it is concluded that the MULTIOPT algorithm is able to yield optimality in any tested case. The cumulative probability distribution illustration yields the same results for finer $\Delta$-granularity.

Figure 2a shows that the valid solutions are bounded by the minimum and maximum values of $c_{i2}$ and $c_{i3}$ as expected. The core area of valid solutions forms a quadrangle. The acmes of this quadrangle correspond to the linear restrictions imposed by equations (14). The vertexes correspond to the extreme mean cost values $\min_{i} / \max_{i} \{c_{i2}\}$, $\min_{i} / \max_{i} \{c_{i3}\}$. These values can be achieved by only one schedule, i.e. broadcasting exclusively the items:

$$i_{1/2} = \arg\min_{i} / \max_{i} \{c_{i2}\}, \ i_{3/4} = \arg\min_{i} / \max_{i} \{c_{i3}\}$$

(19)

Conclusively, MULTIOPT can be effectively used for optimal performance under two concurrent criteria. The study of the cases $C_{i1}$ (query leak rate Vs. copyright costs) and $C_{i3}$ (energy consumption Vs copyright costs) produced identical results.

(a) Simulation results of MULTIOPT and Brute force in the case of two concurrent costs. Energy consumption and copyright costs are examined. MULTIOPT coincides with the performance of the Brute force procedure, at every valid $[C_{i1} \times C_{i2}]$ combination.

(b) Comparison of MULTIOPT versus Brute force for three concurrent costs. Energy consumption $C_{i1}$; query leak rate $C_{i2}$ and copyright costs $C_{i3}$ are considered. MULTIOPT produces optimal results for all $[C_{i1} \times C_{i2} \times C_{i3}]$ combinations.
which is shown in Fig. 3b. The same tetrahedron is produced as the convex hull of the bruteforce-derived solutions and the MULTI OPT derived ones. The extra attribute introduces an additional dimension to the quadrangle of Fig. 2a. As the diversion histogram indicates (Fig. 3a), MULTI OPT produces the same results with the bruteforce process in the 95% of the cases, as in Fig. 2b. MULTI OPT is more efficient in the remaining cases. However, this phenomenon is attributed to the limitation of the bruteforce procedure.

It is worth noting the runtimes required by the MULTI OPT algorithm and the bruteforce procedure. In terms of complexity, bruteforce requires \(500^N\) operations to check all possible combinations of item occurrences \(u_i = 1\ldots500\) for every \(i = 1\ldots N\). MULTI OPT on the other hand, requires \(O(M \cdot N \cdot X)\) operations in the worst case scenario. Thus, the complexity of MULTI OPT is a linear function of the number of items, \(N\), while the complexity of the bruteforce procedure follows a power law. The vast difference in complexity has direct consequences on the typical runtimes of the two processes. For the three criteria optimization case, the MATLAB implementation of MULTI OPT required \(\approx 300\)\(\text{msec}\) on a 3.2GHz core, while the bruteforce procedure took a full week. Furthermore, the granularity of the bruteforce solution is limited into the value set \(u_i = 1\ldots500\), whereas no such restriction applies for MULTI OPT. As a consequence, MULTI OPT achieves strictly equal or better performance than the bruteforce procedure in Fig. 2b and 3a.

C. Discussion on NP-Hardness

The almost perfect results achieved by MULTI OPT do not falsify the proof of the NP-Hardness of the scheduling problem [7]. They do, however, pose a question of whether purely theoretical assumptions have magnified the practical significance of the issue. It has been proven that an optimal schedule is periodic: the interval between two consecutive occurrences of an item must be constant [9]. Therefore, knowledge of the \(u_i\) ratio is at first sufficient for creating an optimal schedule. However, large variations in the size of the data items may hinder periodicity. (E.g. a huge file may not fit in its predefined, periodic positions) [2]. The NP-Hardness stems from the resulting combinatorial problem of finding the optimal item serialization in this case. This issue is not a practical hindrance. In the vast majority of the modern communication systems, large files are divided in much smaller segments or packets, prior to transmission. This approach is known to resolve the aforementioned issue in periodic scheduling as well [2]. For limited item size variations, non-preemptive serializers (e.g. [4]) produced identical, optimal results. Conclusively, having shown that the definition of the optimal \(u_i\) ratios is feasible, the authors claim that the undisputed, theoretical NP-Hardness of the scheduling problem does not pose a significant limitation in practice.

V. Conclusion

The present study reinstated the broadcast cost per data item as a vital input of the periodic scheduling process. A novel scheduler is proposed which can achieve minimal client serving times for any requested mean scheduling cost. The scheduler can also handle any number of additional item attributes concurrently. Simulations showed that the MULTI OPT scheduler can effectively provide optimal serving times in broadcast-based systems, under energy consumption restrictions, copyright cost and information redundancy limitations.

ACKNOWLEDGMENT

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

REFERENCES


