Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9371563, 14 pages
http://dx.doi.org/10.1155/2016/9371563

Research Article

Minimizing the Average Waiting Time of Unequal-Size Data Items in a Mobile Computing Environment

Jen-Ya Wang

Department of Computer Science and Information Management, Hungkuang University, Taichung 43302, Taiwan

Correspondence should be addressed to Jen-Ya Wang; jywang@sunrise.hk.edu.tw

Received 8 November 2015; Revised 19 February 2016; Accepted 29 February 2016

Academic Editor: Lingjie Duan

In a mobile computing environment, waiting time is an important indicator of customer satisfaction. In order to achieve better customer satisfaction and shorter waiting times, we need to overcome different constraints and make the best use of limited resources. In this study, we propose a minimization problem, allocating unequal-size data items to broadcast channels with various bandwidths. The main idea is to solve the problem in the continuous space $\mathbf{R}^n$. First, we map the discrete optimization problem from $\mathbf{Z}^n$ to $\mathbf{R}^n$. Second, the mapped problem is solved in $\mathbf{R}^n$ optimally. Finally, we map the optimal solution from $\mathbf{R}^n$ back to $\mathbf{Z}^n$. With the theoretical analyses, we can ensure the solution quality and execution speed. Computational experiments show that the proposed algorithm performs well. The worst mean relative error can be reduced to 0.353 for data items with a mean size of 100. Moreover, almost all the near-optimal solutions can be obtained within 1 millisecond, even for $N = 500$, where N is the number of data items, that is, the problem size.

1. Introduction

Broadcasting is an efficient mechanism to transmit information in a mobile computing environment. Popular messages (e.g., weather reports) or instant information (e.g., stock quotes) can be widely disseminated via the broadcast mechanism. This success is mainly due to the high bandwidths of downlinks, which are used for the transmission of data items from a broadcast server to unlimited mobile users. Note that the bandwidths of channels and the sizes of data items might be unequal. The simplest strategy is to equally allocate data items to all channels, or load balance, but it is not the best way to reduce waiting time. Consequently, sophisticated broadcast scheduling or data partition algorithms are called for.

Waiting time, or expected delay, is an important indicator for measuring broadcast performance, for the waiting times of mobile users directly influence their customer satisfaction [1–5]. For example, Chien and Lin [3] found that customers' waiting experiences may negatively affect their attitudes towards a given service. Moreover, improving the service implies the improvement of user experiences. Therefore, a good broadcast mechanism that is able to reduce waiting time can achieve better customer satisfaction.

Let us observe the simplest form of such problems, that is, $\sum_i[(\sum_j a_j)(\sum_j 1)]$. These problems feature single-item queries, multiple channels, and skewed data preferences. Several equal-bandwidth broadcast channels and multiple equal-size data items need to be broadcast over multiple channels periodically. In a mobile computing environment, users can download their desired items via their mobile devices, such as smartphones. Imagine that we allocate a few popular items to a channel, that is, a short cycle length, and other ordinary items to another channel, that is, a long cycle length. Most users can download popular items in a short time, without a long wait. Clearly, access probability and cycle length directly influence broadcast scheduling. Note the unbalanced workloads of these channels (i.e., the different amounts of data) in an optimal allocation. Once the optimal schedule is found, we can achieve shorter waiting times and better customer satisfaction. However, determining optimal schedules requires much execution time. That is, the time complexity excludes the related optimization algorithms from practical use.

Now we consider some complicated forms of such problems, that is, $\sum_i[(\sum_j a_j)(\sum_j b_j)]$. These problems may have unequal-size data items or various bandwidths. There

are still multiple channels and skewed preferences for data items. For example, in [6], the authors assumed that the bandwidths were different. This consideration makes this problem more difficult. In [7], the authors considered another problem, in which the sizes of data items are unequal. This assumption makes the problem more flexible and practical.

In recent years, various other forms have also been studied. In [8], mobile users could download multiple items at a time by sending a simple query. The authors assumed that two queries might have some items in common. To shorten the broadcast cycle length, the duplicate items were centralized and allocated to the same channel. In [9], video was broadcast on a single channel, so the data size was variable. In [10], mobile users could download multiple items in a multichannel environment. However, the wireless links were unreliable, so disconnections occurred frequently. In that study, reducing the waiting time was not the first priority. Instead, the authors aimed to minimize the deadline miss rate. All of the above considerations make the problems more complicated. Obviously, these problems cannot be solved easily or optimally when the problem size is large, so we need some more efficient algorithms to deal with these problems.

Such problems are usually time-consuming or even NP-hard, so several intuitive or metaheuristic algorithms have been proposed. However, traditional algorithms have some shortcomings. For example, although dynamic programming [7, 12] and branch-and-bound algorithms [13, 14] can provide optimal solutions, they are time-consuming and cannot be applied to large problem instances; for example, $N = 500$. On the other hand, some metaheuristic algorithms [8] or greedy approaches [12, 15, 16] generate solutions very quickly. Nevertheless, their solution qualities are not stable. The reason is that their searches are done by random walks in their solution spaces. For the same problem instance, multiple executions of an identical algorithm can generate different solutions. Moreover, as in the case of tabu search, a great amount of memory is needed for keeping track of past experiences. Such algorithms are also unsuitable for large problem instances.

In this study, we consider a waiting time minimization problem (abbreviated as WTM). To make this problem more flexible and practical, meaning that unequal bandwidths and various data sizes are allowed, we propose a linearly convergent algorithm based on a steepest decent technique. First, WTM is mapped from $\mathbf{Z}^n$ (the discretized space) to $\mathbf{R}^n$ (the continuous space). Next, WTM$'$ in $\mathbf{R}^n$ is solved optimally in linear time. Finally, the optimal solution is mapped from $\mathbf{R}^n$ back to $\mathbf{Z}^n$.

The rest of this study is organized as follows. Section 2 gives the formal definition of the proposed problem. Section 3 establishes the theoretical basis for the problem. Section 4 presents the linearly convergent algorithm. This study is compared with past research in Section 5. In Section 6, computational experiments are conducted to evaluate the performance of the proposed algorithm. Finally, conclusions are drawn in Section 7.

2. Problem Formulation

The waiting time minimization problem (WTM) is formulated as follows. There is a database $\mathbf{D} = \{d_1, d_2, \ldots, d_N\}$ to be broadcast in a mobile computing environment, where d_j is the jth data item for $j = 1, 2, \ldots, N$. Let p_j and λ_j denote the access probability and the size of d_j, respectively. The total amount of data is $\Lambda (= \sum_{j=1}^{N} \lambda_j)$. Assume that the access pattern S, that is, the sequence of (p_j, λ_j), is given in advance. Assume that a broadcast server is equipped with C channels, numbered from 1 to C. We let the bandwidth of channel i be w_i. Without loss of generality, assume that $w_i \geq w_j$ for all $i < j$. Each channel is divided into time slices of equal size, called buckets. We need to partition $\mathbf{D}$ into C parts based on the access pattern S and assign each part to one of the C channels, one item to several consecutive buckets. Then C broadcast programs are formed, and each of them will be broadcast cyclically. The average waiting time is defined as the average amount of waiting time spent by each user before he/she receives the desired data item. Finally, the objective is to minimize the average waiting time of the C programs under the above assumptions. The objective function can be written as $\min \sum_{i=1}^{C} [(\sum_{d_j \in M_i} \lambda_j)(\sum_{d_j \in M_i} p_j)/w_i]$, where $d_j \in M_i$ means d_j is allocated to channel i or machine i.

Three properties regarding WTM are discussed as follows. First, for each single channel i, its corresponding broadcast program cycle length is $\Lambda_i = \sum_{d_j \in M_i} \lambda_j$. Then the expected waiting time in receiving d_j on the channel is $(0.5\Lambda_i + \lambda_j)/w_i$. If Λ_i is much larger than λ_j, the expected waiting time can be simplified to $0.5\Lambda_i/w_i$. Namely, the position order of data items makes no difference on the final result, that is, the average waiting time. Second, the average waiting time can be determined only by data partition. That is, the average waiting time is the weighted sum of the average waiting time on each channel, that is, $0.5 \sum_{i=1}^{C} (\Lambda_i \sum_{d_j \in M_i} p_j)/w_i$. Third, such minimization problems are time-consuming or even NP-hard [6, 16, 17]. Consequently, some efficient minimization algorithms are called for. With the above observation, we can define proper objective functions in the next section.

3. Theoretical Basis

In this section, a steepest decent technique is employed to solve the problem. First, we map the WTM problem from $\mathbf{Z}^n$ to a new problem WTM$'$ in $\mathbf{R}^n$. Second, we employ a steepest decent technique [11, 18] to obtain the optimal solution to WTM$'$ in $\mathbf{R}^n$. Finally, we map the optimal solution from $\mathbf{R}^n$ back to $\mathbf{Z}^n$. The main idea behind the transformation is to improve the solution quality and execution speed. Although WTM can be solved optimally by some optimization algorithms (e.g., a branch-and-bound algorithm) in $\mathbf{Z}^n$, it is too time-consuming when N is large. On the other hand, though some metaheuristic algorithms (e.g., GA) are able to provide instant solutions in $\mathbf{Z}^n$, their solution quality is not guaranteed. For the same problem instance, they may provide different solutions. Consequently, instead of directly optimizing WTM in $\mathbf{Z}^n$, we map WTM to $\mathbf{R}^n$ and take

advantage of the linear convergence and optimality of the gradient-based technique in $\mathbf{R}^n$.

The parameters used in this study are summarized in Parameters at the end of the paper. Parameters N, C, and S are defined earlier. Parameters $S0$ and $S1$ are the access patterns used in different spaces, that is, $\mathbf{Z}^n$ and $\mathbf{R}^n$. We also need two cumulative functions, $P(j)$, $Q(j)$, and two interpolating functions, $F(x)$, $H(x)$, to define the objective functions $\alpha_S(\mathbf{n})$ in $\mathbf{R}^n$ and $\beta_S(\mathbf{x})$ in $\mathbf{Z}^n$, respectively. Moreover, $\mathbf{n} \in \mathbf{Z}^n$ and $\mathbf{x} \in \mathbf{R}^n$ are partition vectors or position vectors.

3.1. Mapping WTM from $\mathbf{Z}^n$ to $\mathbf{R}^n$. First, two objective functions for both WTM and WTM$'$ are defined. Then we show how to map WTM from $\mathbf{Z}^n$ to WTM$'$ in $\mathbf{R}^n$. Finally, we prove that the geometric properties, such as concavity, of both WTM and WTM$'$ are similar. These proofs ensure that both solution spaces are close to each other.

The relationship between WTM and WTM$'$ is similar to that between the 0-1 knapsack problem and the fractional knapsack problem [19]. If we can solve the problems in $\mathbf{R}^n$ optimally, then the rounded solutions mapped back to $\mathbf{Z}^n$ are therefore near-optimal solutions to the original problems. To show this relationship, two proper objective functions play an important role. Namely, the objective functions of WTM and WTM$'$ must resemble each other. Once the two similar objective functions are determined, we can claim that the optimal solution of WTM$'$ is very close to that of WTM.

Definition 1 helps us to transform the *data* partition problem into a *sequence* partition problem. Since the position order of an access pattern will lead to different results, we need the following definition to help us to ensure the optimality of WTM and WTM$'$.

Definition 1. Given an optimal solution, the C programs can be concatenated into an optimal program. Let $S0$ denote the sequence of (p_i, λ_i) that has the same order as the optimal program for WTM. Similarly, let $S1$ denote the sequence of (p_i, λ_i) that has the same order as the optimal program for WTM$'$.

WTM and WTM$'$ have different preferences for the order of access pattern. Consider the relationship between the 0-1 knapsack problem and the fractional knapsack problem again. For the fractional knapsack problem, if we take the items of greatest unit value one by one in a preemptive manner, we can easily achieve the maximum objective cost [19]. Thus we try to solve the continuous-case WTM$'$ by sorting the items d_i in descending order of (p_i/λ_i). That is, the sequence $S1$ is obtained by sorting (p_i/λ_i) in nonincreasing order. On the other hand, for the 0-1 knapsack problem, we may achieve the optimal solution but sacrifice some valuable items because they are too big to fit the knapsack. Clearly, the optimal item partition of the 0-1 knapsack problem cannot be solved by using simple sorting rules. That is, when we reduce an item partition problem to a sequence partition problem, it is difficult to determine the optimal sequence for partition. Consequently, we do not aim to find the optimal sequence $S0$ for WTM. Instead, we just use $S1$ for WTM$'$ to simulate $S0$.

Definitions 2 and 3 introduce two cumulative functions regarding the access pattern S. With the two cumulative functions, we can define a new objective function in $\mathbf{R}^n$ in a more comprehensive way for later transformation.

Definition 2. Given S, the function $P(j)$ of cumulative probability is defined by

$$P(j) = \begin{cases} \sum_{k=1}^{j} p_k, & 1 \le j \le N, \\ 0, & j = 0. \end{cases} \tag{1}$$

Similarly, we define another function to express the cycle length of a broadcast program. The function $Q(j)$ is defined as follows.

Definition 3. Given S, the function $Q(j)$ of the cumulative data size is defined by

$$Q(j) = \begin{cases} \sum_{k=1}^{j} \lambda_k, & 1 \le j \le N, \\ 0, & j = 0. \end{cases} \tag{2}$$

Now we redefine the objective function for the original problem WTM in $\mathbf{Z}^n$. The original objective function $\sum_{i=1}^{C}[(\sum_{d_j \in M_i} \lambda_j)(\sum_{d_j \in M_i} p_j)/w_i]$ is defined from the viewpoint of *data* partition, whereas the new objective function is formulated from the viewpoint of *sequence* partition. With the cumulative functions $P(j)$ and $Q(j)$, we can determine a proper sequence in advance and then perform partition on the sequence. For simplicity, the leading coefficient of expected waiting time of each channel, 0.5, is omitted in the rest of this study.

Definition 4. Given S and two constants $n_0 = 0$, $n_C = N$ for any column vector $\mathbf{n} = [n_1, n_2, \ldots, n_{C-1}]^t \in \{1, 2, \ldots, N\}^{C-1}$ with $n_{i-1} \le n_i$, the objective function $\alpha_S(\mathbf{n})$ of WTM is defined by

$$\alpha_S(\mathbf{n}) = \sum_{i=1}^{C} \frac{(Q(n_i) - Q(n_{i-1}))(P(n_i) - P(n_{i-1}))}{w_i}, \tag{3}$$

where w_i is the bandwidth of channel i.

Note that data items numbered $n_{i-1} + 1, n_{i-1} + 2, \ldots, n_i$ are allocated to channel i. Therefore, the access probability of channel i is $P(n_i) - P(n_{i-1})$ and the program cycle length of the channel is $Q(n_i) - Q(n_{i-1})$.

An interpolating function $F(x)$ regarding access probability is defined for mapping WTM from $\mathbf{Z}^n$ to $\mathbf{R}^n$. To preserve the geometric properties, we interpolate the $N + 1$ points $(j, P(j))$, $j = 0, 1, \ldots, N$, by using N separate line segments. The interpolating function $F(x)$ is defined as follows.

Definition 5. For $j = 1, 2, \ldots, N$, consider the jth interval $[j-1, j]$ and let the straight line $f_j(x)$ pass through the two points

$(j-1, P(j-1))$ and $(j, P(j))$. The interpolating function $F(x)$ is given by

$$F(x) = \begin{cases} f_1(x), & \text{if } x \in [0,1), \\ f_2(x), & \text{if } x \in [1,2), \\ \vdots & \\ f_{N-1}(x), & \text{if } x \in [N-2, N-1), \\ f_N(x), & \text{if } x \in [N-1, N]. \end{cases} \tag{4}$$

An interpolating function $H(x)$ regarding program cycle length is also defined as follows. We let it interpolate the $N+1$ points $(j, Q(j))$, $j = 0, 1, \ldots, N$, by using N separate line segments.

Definition 6. For $j = 1, 2, \ldots, N$, consider the jth interval $[j-1, j]$ and let the straight line $h_j(x)$ pass through the two points $(j-1, Q(j-1))$ and $(j, Q(j))$. The interpolating function $H(x)$ is given by

$$H(x) = \begin{cases} h_1(x), & \text{if } x \in [0,1), \\ h_2(x), & \text{if } x \in [1,2), \\ \vdots & \\ h_{N-1}(x), & \text{if } x \in [N-2, N-1), \\ h_N(x), & \text{if } x \in [N-1, N]. \end{cases} \tag{5}$$

Just as we define the objective function $\alpha_S(\mathbf{n})$ for WTM in $\mathbf{Z}^n$, we define a dummy objective function $\beta_S(\mathbf{x})$ for WTM$'$ in $\mathbf{R}^n$ as follows.

Definition 7. Given S and two constants $x_0 = 0$, $x_C = N$, for any position vector $\mathbf{x} = [x_1, x_2, \ldots, x_{C-1}]^t \in (0, N)^{C-1}$ with $x_{i-1} < x_i$, the objective function $\beta_S(\mathbf{x})$ of WTM$'$ is defined by

$$\beta_S(\mathbf{x}) = \sum_{i=1}^{C} \frac{(H(x_i) - H(x_{i-1}))(F(x_i) - F(x_{i-1}))}{w_i}, \tag{6}$$

where $F(x), Q(x)$ are the interpolating functions and w_i is the bandwidth of channel i.

Since $\alpha_S(\mathbf{n})$ and $\beta_S(\mathbf{x})$ have the same objective costs at all grid points, the solution spaces of WTM and WTM$'$ are of similar geometric properties, for example, slope, extreme, and concavity. Lemma 8 shows that both functions agree with each other at all grid points.

Lemma 8. *Given S, for any $\mathbf{n} \in \{1, 2, \ldots, N\}^{C-1}$, $\alpha_S(\mathbf{n}) = \beta_S(\mathbf{n})$.*

Proof. Note that $f_{j+1}(j) = P(j)$ and $h_{j+1}(j) = Q(j)$ for all $j = 1, 2, \ldots, N$. Then

$$\begin{aligned} \beta_S(\mathbf{n}) &= \sum_{i=1}^{C} \frac{(H(n_i) - H(n_{i-1}))(F(n_i) - F(n_{i-1}))}{w_i} \\ &= \sum_{i=1}^{C} \frac{(H(n_i) - H(n_{i-1}))\left(f_{n_i+1}(n_i) - f_{n_{i-1}+1}(n_{i-1})\right)}{w_i} \\ &\qquad \text{(by Definition 5)} \\ &= \sum_{i=1}^{C} \frac{\left(h_{n_i+1}(n_i) - h_{n_{i-1}+1}(n_{i-1})\right)\left(f_{n_i+1}(n_i) - f_{n_{i-1}+1}(n_{i-1})\right)}{w_i} \\ &\qquad \text{(by Definition 6)} \\ &= \sum_{i=1}^{C} \frac{(Q(n_i) - Q(n_{i-1}))(P(n_i) - P(n_{i-1}))}{w_i} \\ &= \alpha_S(\mathbf{n}). \end{aligned} \tag{7}$$

The proof is complete. □

By Lemma 8, $\alpha_S(\mathbf{n})$ and $\beta_S(\mathbf{x})$ have the same function values at grid points. That is, they have similar geometric properties. Therefore, the optimal solution to WTM in $\mathbf{Z}^n$ is close to that of WTM$'$ in $\mathbf{R}^n$. The optimal solutions to WTM and WTM$'$ are also close to each other.

3.2. Optimal Solution $\mathbf{x}^*$ to WTM$'$. In this subsection, we solve WTM$'$ optimally and obtain the optimal solution $\mathbf{x}^*$ in $\mathbf{R}^n$. First, we introduce the concept of gradient. Then the optimality and convergence speed are discussed.

The steepest decent technique we employed is based on the concept of the gradient [11]. Unlike other metaheuristic algorithms, the steepest decent technique always converges at the global minimum instead of piecing several local minimums. Moreover, this steepest decent technique converges in n dimensions in linear time instead of performing meaningless random walks. The notation of the gradient is defined as follows.

Definition 9. Let g: $\mathbf{R}^n \to \mathbf{R}$ be a continuous differentiable multivariable function. The gradient of g at $\mathbf{x} = [x_1, x_2, \ldots, x_n]^t$ is denoted by $\nabla g(\mathbf{x})$ and defined by

$$\nabla g(\mathbf{x}) = \left[\frac{\partial g(\mathbf{x})}{\partial x_1}, \frac{\partial g(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial g(\mathbf{x})}{\partial x_n}\right]^t. \tag{8}$$

Because $F(x)$ is not differentiable at $x = 0, 1, 2, \ldots, N-1$, we remedy this shortcoming by improper limit. The ith element of $\nabla g(\mathbf{x})$ is modified slightly and redefined as

$$\frac{\partial g(\mathbf{x})}{\partial x_i} = \begin{cases} \lim_{h \to 0^+} \dfrac{g(x_1, x_2, \ldots, x_i + h, \ldots, x_{C-1}) - g(x_1, x_2, \ldots, x_i, \ldots, x_{C-1})}{h}, & \text{if } x_i \text{ is an integer}, \\ \lim_{h \to 0} \dfrac{g(x_1, x_2, \ldots, x_i + h, \ldots, x_{C-1}) - g(x_1, x_2, \ldots, x_i, \ldots, x_{C-1})}{h}, & \text{otherwise}. \end{cases} \tag{9}$$

Note that $F(x)$ and $H(x)$ are therefore right-hand differentiable for $x \in \{0, 1, 2, \ldots, N-1\}$. Thus, the gradient of $\beta_S(\mathbf{x})$ can be obtained for any $\mathbf{x} \in (0, N)^{C-1}$, and the ith element of $\nabla\beta_S(\mathbf{x})$ becomes

$$\begin{aligned}
\frac{\partial \beta_S(\mathbf{x})}{\partial x_i} &= \frac{1}{w_i}\left[H'(x_i)\left(F(x_i) - F(x_{i-1})\right) + \left(H(x_i)\right.\right. \\
&\quad \left.\left. - H(x_{i-1})\right)F'(x_i)\right] - \frac{1}{w_{i+1}}\left[H'(x_i)\left(F(x_{i+1})\right.\right. \\
&\quad \left.\left. - F(x_i)\right) + \left(H(x_{i+1}) - H(x_i)\right)F'(x_i)\right] \\
&= \frac{1}{w_i}\left[\lambda_b\left(F(x_i) - F(x_{i-1})\right) + \left(H(x_i) - H(x_{i-1})\right)\right. \\
&\quad \left. \cdot\, p_b\right] - \frac{1}{w_{i+1}}\left[\lambda_b\left(F(x_{i+1}) - F(x_i)\right) + \left(H(x_{i+1})\right.\right. \\
&\quad \left.\left. - H(x_i)\right)p_b\right] = \frac{1}{w_i}\left[\lambda_b\left(\left(P(b) + (x_i - b)\,p_{b+1}\right)\right.\right. \\
&\quad - \left(P(a) + (x_{i-1} - a)\,p_{a+1}\right)\Big) \\
&\quad + \left(\left(Q(b) + (x_i - b)\,\lambda_{b+1}\right)\right. \\
&\quad \left.\left. - \left(Q(a) + (x_{i-1} - a)\,\lambda_{a+1}\right)\right)p_b\right] \\
&\quad - \frac{1}{w_{i+1}}\left[\lambda_b\left(\left(P(c) + (x_{i+1} - c)\,p_{c+1}\right)\right) - \left(P(b)\right.\right. \\
&\quad \left.\left. + (x_i - b)\,p_{b+1}\right) + \left(\left(Q(c) + (x_{i+1} - c)\,\lambda_{c+1}\right)\right. \\
&\quad \left.\left. - \left(Q(b) + (x_i - b)\,\lambda_{b+1}\right)\right)p_b\right]
\end{aligned} \tag{10}$$

for three integers $a \le b \le c$ with $x_{i-1} \in [a, a+1)$, $x_i \in [b, b+1)$, and $x_{i+1} \in [c, c+1)$. More specifically, when programming, we do not need to find the derivative of $H(x_i)$ for obtaining the corresponding slope. Instead, we obtain the slope λ_b by letting $\lfloor x_i \rfloor = b$. Hence, we eliminate annoying differentiation procedures in the proposed algorithm.

Similar algorithms are found in [2, 11, 20, 21]. We can modify them slightly in order to obtain the optimal solution $\mathbf{x}^*$ in $\mathbf{R}^n$. The details of the algorithm will be presented in the next section. Here, we show only the basic steps of the steepest decent technique as follows:

(1) Evaluate $\beta_S(\mathbf{x})$ at an initial position $\mathbf{x}^{(0)}$.

(2) Determine the steepest decent direction from $\mathbf{x}^{(0)}$ that results in a decrease in the value of β_S.

(3) Move an appropriate amount δ (i.e., step size) in this direction, and the new position is $\mathbf{x}^{(1)}$.

(4) Set $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \delta\nabla\beta_S(\mathbf{x}^{(0)})$.

(5) Repeat steps (1) through (4) until the optimal solution $\mathbf{x}^*$ is obtained.

In order to implement the algorithm easily, we reduce the equation

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \delta\nabla\beta_S\left(\mathbf{x}^{(0)}\right) \tag{11}$$

to a single variable function

$$v(\delta) = \beta_S\left(\mathbf{x}^{(0)} - \delta\nabla\beta_S\left(\mathbf{x}^{(0)}\right)\right). \tag{12}$$

Note that the value δ_0 that minimizes $v(\delta)$ is also the value needed for (11). Because the root-finding process in (12) requires much execution time, Burden and Faires [11] employed a quadratic polynomial to interpolate $v(\delta)$ in order to accelerate the root-finding process. The details regarding the quadratic polynomial will be shown in the next section.

The proposed algorithm converges linearly and the proofs of convergence are omitted. Readers can refer to [11, 18]. Even if this algorithm converges rapidly, an accurate initial solution can accelerate the convergence speed more. The following definition and lemma help us to choose an accurate initial solution.

Definition 10. Let $x_0 = 0$, $x_C = N$ be two constants. For any position vector $\mathbf{x} = [x_1, x_2, \ldots, x_{C-1}]^t \in (0, N)^{C-1}$, the length of the ith interval $[x_{i-1}, x_i]$ is denoted by Δx_i and defined by

$$\Delta x_i = x_i - x_{i-1} \tag{13}$$

for $i = 1, 2, \ldots, C$. Likewise, the difference of H caused by Δx_i is defined by

$$\Delta H_i = H(x_i) - H(x_{i-1}) \tag{14}$$

for $i = 1, 2, \ldots, C$.

The following lemma shows how to obtain an accurate initial solution by determining the elements of a position vector or partition vector $\mathbf{x}$.

Lemma 11. *If the optimal solution $\mathbf{x}^*$ to the dummy objective function $\beta_{S1}(\mathbf{x})$ is given, then $\Delta H_i^*/w_i \le \Delta H_{i+1}^*/w_{i+1}$ for $i = 1, 2, \ldots, C-1$.*

Proof. We show it by contradiction. Suppose $\Delta H_i^*/w_i > \Delta H_{i+1}^*/w_{i+1}$ for some i. Since $\mathbf{x}^*$ is the optimal solution to $\beta_{S1}(\mathbf{x})$, the sum of products

$$\begin{aligned}
S^* &= \frac{\left[\left(H(x_i^*) - H(x_{i-1}^*)\right)\left(F(x_i^*) - F(x_{i-1}^*)\right)\right]}{w_i} \\
&\quad + \frac{\left[\left(H(x_{i+1}^*) - H(x_i^*)\right)\left(F(x_{i+1}^*) - F(x_i^*)\right)\right]}{w_{i+1}}
\end{aligned} \tag{15}$$

should be the minimal in the interval $[x_{i-1}^*, x_{i+1}^*]$. On the other hand, since $\Delta H_i^*/w_i > \Delta H_{i+1}^*/w_{i+1}$, there exists a number $x_i < x_i^*$ such that $(H(x_i) - H(x_{i-1}^*))/w_i = (H(x_{i+1}^*) - H(x_i))/w_{i+1}$. We claim that

$$\begin{aligned}
S &= \frac{\left[\left(H(x_i) - H(x_{i-1}^*)\right)\left(F(x_i) - F(x_{i-1}^*)\right)\right]}{w_i} \\
&\quad + \frac{\left[\left(H(x_{i+1}^*) - H(x_i)\right)\left(F(x_{i+1}^*) - F(x_i)\right)\right]}{w_{i+1}} \\
&< S^*.
\end{aligned} \tag{16}$$

Let $H(x_i^*) - H(x_{i-1}^*) = a$, $F(x_i^*) - F(x_{i-1}^*) = A$, $H(x_{i+1}^*) - H(x_i^*) = b$, $F(x_{i+1}^*) - F(x_i^*) = B$, $H(x_i^*) - H(x_i) = \delta$, $F(x_i^*) - F(x_i) = \Delta$, and $(H(x_i) - H(x_{i-1}^*))/w_i = (H(x_{i+1}^*) - H(x_{i-1}))/w_{i+1} = \pi$. Since the sequence $S1$ is sorted by (p_i/λ_i) in descending order, $\Delta H_i^*/w_i > \Delta H_{i+1}^*/w_{i+1}$ implies that $(F(x_i^*) - F(x_{i-1}^*))/w_i > (F(x_{i+1}^*) - F(x_i^*))/w_{i+1}$. That is, if $a/w_i > b/w_{i+1}$, then $A/w_i > B/w_{i+1}$. Then

$$\begin{aligned} S &= \frac{1}{w_i}(a-\delta)(A-\Delta) + \frac{1}{w_{i+1}}(b+\delta)(B+\Delta) \\ &= \pi(A-\Delta) + \pi(B+\Delta) = \pi A + \pi B \\ &= \frac{1}{w_i}(a-\delta)A + \frac{1}{w_{i+1}}(b+\delta)B \\ &= \frac{aA}{w_i} + \frac{bB}{w_{i+1}} - \frac{\delta A}{w_i} + \frac{\delta B}{w_{i+1}} < \frac{aA}{w_i} + \frac{bB}{w_{i+1}} = S^*. \end{aligned} \tag{17}$$

It contradicts that $\mathbf{x}^*$ is the optimal solution. The proof is complete. □

By the above lemma, the proposed algorithm is therefore able to start from a better initial position $\mathbf{x}^{(0)}$. We find $x_1^{(0)}, x_2^{(0)}, \ldots$, and $x_{C-1}^{(0)}$ such that $\Delta H_i^*/w_i \le \Delta H_{i+1}^*/w_{i+1} = \Lambda/W$, where $W = \sum_{i=1}^{C} w_i$. In the next section, we set $\mathbf{x}^{(0)} = [x_1^{(0)}, x_2^{(0)}, \ldots, x_{C-1}^{(0)}]^t$ instead of randomly choosing $\mathbf{x}^{(0)}$. This initial position will also enhance the convergence speed.

Note that the partition vector $\mathbf{x}^*$ is a *global* minimizer of WTM′. We prove this property by showing that $\beta_{S1}(\mathbf{x})$ is concave upward for all $\mathbf{x} \in (0, N)^{C-1}$. Namely, for any initial vector, the algorithm converges to the global minimum.

Lemma 12. *Let the algorithm converge locally at some* $\mathbf{x}^* \in (0, N)^{C-1}$. *Then* $\mathbf{x}^*$ *is the global optimal solution to WTM′.*

Proof. We prove the property by showing that, for any $\mathbf{x} \in (0, N)^{C-1}$, $\beta_{S1}(\mathbf{x})$ is concave upward. From (10), we get

$$\begin{aligned} \frac{\partial \beta_{S1}(\mathbf{x})}{\partial x_i} &= \frac{1}{w_i}\left[H'(x_i)(F(x_i) - F(x_{i-1})) \right. \\ &\quad \left. + (H(x_i) - H(x_{i-1}))F'(x_i)\right] \\ &\quad - \frac{1}{w_{i+1}}\left[H'(x_i)(F(x_{i+1}) - F(x_i)) \right. \\ &\quad \left. + (H(x_{i+1}) - H(x_i))F'(x_i)\right], \end{aligned} \tag{18}$$

for $i = 1, 2, \ldots, C-1$. Note that $x_0 = 0$ and $x_C = N$ are two constants. Then the second partial derivative is

$$\begin{aligned} \frac{\partial \beta_{S1}^2(\mathbf{x})}{\partial x_i^2} &= \frac{1}{w_i}\left[H''(x_i)(F(x_i) - F(x_{i-1})) \right. \\ &\quad \left. + 2H'(x_i)F'(x_i) + (H(x_i) - H(x_{i-1}))F''(x_i)\right] \\ &\quad - \frac{1}{w_{i+1}}\left[H''(x_i)(F(x_{i+1}) - F(x_i)) \right. \\ &\quad \left. - 2H'(x_i)H'(x_i) + (H(x_{i+1}) - H(x_i))F''(x_i)\right] \\ &= \frac{2}{w_i}H'(x_i)F'(x_i) + \frac{2}{w_{i+1}}H'(x_i)H'(x_i). \end{aligned} \tag{19}$$

Because $F(x)$ interpolates the strictly increasing function $P(i)$, it is clear that $F'(x) > 0$. We have $F''(x) = 0$, since $F(x)$ is composed of straight lines. Similarly, $H(x)$ has the same properties. Therefore, $\partial \beta_{S1}^2(\mathbf{x})/\partial x_i^2 > 0$ for all $\mathbf{x} \in (0, N)^{C-1}$. The proof is complete. □

3.3. Near-Optimal Solution $\mathbf{n}^\dagger$ *to WTM.* We show how to obtain a near-optimal solution of WTM in this subsection. First, we remap the optimal solution $\mathbf{x}^*$ in $\mathbf{R}^n$ back to $\mathbf{Z}^n$ by rounding off all elements of $\mathbf{x}^*$, and a rounded solution $\mathbf{n}^\dagger \in \{1, 2, \ldots, N\}^{C-1}$ is obtained. The following lemma shows the magnitude order of $\beta_{S1}(\mathbf{x}^*)$, $\alpha_{S0}(\mathbf{n}^*)$, $\alpha_{S1}(\mathbf{n}^\#)$, and $\alpha_{S1}(\mathbf{n}^\dagger)$, where $\mathbf{n}^\# \in \{1, 2, \ldots, N\}^{C-1}$ minimizes $\alpha_{S1}(\mathbf{n})$ and $\mathbf{n}^* \in \{1, 2, \ldots, N\}^{C-1}$ minimizes $\alpha_{S0}(\mathbf{n})$.

Lemma 13. *Let* $\mathbf{n}^\dagger \in \{1, 2, \ldots, N\}^{C-1}$ *be the rounded resulting solution to WTM. Then* $\beta_{S1}(\mathbf{x}^*)$ *and* $\alpha_{S1}(\mathbf{n}^\dagger)$ *are two bounds of* $\alpha_{S0}(\mathbf{n}^*)$ *with*

$$\beta_{S1}(\mathbf{x}^*) \le \alpha_{S0}(\mathbf{n}^*) \le \alpha_{S1}(\mathbf{n}^\#) \le \alpha_{S1}(\mathbf{n}^\dagger), \tag{20}$$

and the Euclidean distance between $\mathbf{x}^*$ *and* $\mathbf{n}^\dagger$ *is*

$$\|\mathbf{x}^* - \mathbf{n}^\dagger\|_2 \le 0.5\sqrt{C-1}. \tag{21}$$

Proof. Since $\mathbf{n}^\#$ minimizes $\alpha_{S1}(\mathbf{n})$, $\alpha_{S1}(\mathbf{n}^\#) \le \alpha_{S1}(\mathbf{n}^\dagger)$. On the other hand, we need to show $\alpha_{S0}(\mathbf{n}^*) \le \alpha_{S1}(\mathbf{n}^\#)$. Note that $S0$ is the optimal sequence to the discrete-case problem WTM and that it is difficult to determine. We just assume that $\mathbf{n}^*$ is the optimal solution to $\alpha_{S0}(\mathbf{n})$. Since $S1$ is obtained by some simple sorting rule and dedicated to the continuous-case problem WTM′, the sequence $S0$ is more suitable for the discrete-case objective function, $\alpha_S(\mathbf{n})$, than $S1$. Therefore, we obtain $\alpha_{S0}(\mathbf{n}^*) \le \alpha_{S1}(\mathbf{n}^\#)$.

We show $\beta_{S1}(\mathbf{x}^*) \le \alpha_{S0}(\mathbf{n}^*)$ by contradiction. Suppose $\alpha_{S0}(\mathbf{n}^*) < \beta_{S1}(\mathbf{x}^*)$. On the other hand, by Lemma 8, $\beta_{S0}(\mathbf{n}^*) = \alpha_{S0}(\mathbf{n}^*)$, since $\mathbf{n}^* \in \{1, 2, \ldots, N\}^{C-1}$. Therefore, $\beta_{S0}(\mathbf{n}^*) = \alpha_{S0}(\mathbf{n}^*) < \beta_{S1}(\mathbf{x}^*)$. However, $\beta_{S1}(\mathbf{x}^*)$ is the optimal solution to WTM′. It is a contradiction.

Since each $n_i^\dagger$ is rounded from x_i^*, it is obvious that $|x_i^* - n_i^\dagger| \le 0.5$ and thus $\|\mathbf{x}^* - \mathbf{n}^\dagger\|_2 \le 0.5\sqrt{C-1}$. The proof is complete. □

By Lemma 13, we know that $\alpha_{S0}(\mathbf{n}^*)$ is bounded by $\beta_{S1}(\mathbf{x}^*)$ and $\alpha_{S1}(\mathbf{n}^\dagger)$ or $\beta_{S1}(\mathbf{n}^\dagger)$, since $\alpha_{S1}(\mathbf{n}^\dagger) = \beta_{S1}(\mathbf{n}^\dagger)$ by Lemma 8. This guarantees that the proposed algorithm will output near-optimal broadcast programs.

4. Proposed Algorithm

Algorithm 1 shows the proposed algorithm GRA. In the first step, we prepare the cumulative functions $P(i)$ and

Procedure **GRA** (C, **D**, $S1$, TOL, $K_{\max}$)
INPUT: the number of channels C, the database **D**, the sorted access pattern $S1$,
the tolerance TOL, the maximal number of iterations $K_{\max}$.
OUTPUT: the near-optimal solution $\mathbf{n}^{\dagger}$ to WTM.
Step 1 Calculate $P(i)$, $Q(i)$, and construct corresponding $F(x)$, $H(x)$;
Construct the dummy objective function $\beta_{S1}(\mathbf{x})$;
Set $k = 1$; $\mathbf{x} = [x_1^{(0)}, x_2^{(0)}, \ldots, x_{C-1}^{(0)}]^t$.
Step 2 **While** ($k \le K_{\max}$) **do** Steps 3–15.
Step 3 Set $\beta_1 = \beta_{S1}(\mathbf{x})$; $\mathbf{z} = \nabla\beta_{S1}(\mathbf{x})$; $z_0 = \|\mathbf{z}\|_2$.
//Note that $\|\mathbf{z}\|_2$ is the Euclidean distance of $\mathbf{z}$.
Step 4 **If** ($z_0 = 0$) **then**
Output "$x_1, x_2, \ldots, x_{C-1}, \beta_1$", "Zero gradient!";
Stop.
Step 5 Set $\mathbf{z} = \mathbf{z}/z_0$; $\delta_1 = 0$; $\delta_3 = 1$; $\beta_3 = \beta_{S1}(\mathbf{x} - \delta_3\mathbf{z})$.
Step 6 **While** ($|\beta_3| \ge |\beta_1|$) **do** Steps 7 and 8.
Step 7 Set $\delta_3 = \delta_3/2$; $\beta_3 = \beta_{S1}(\mathbf{x} - \delta_3\mathbf{z})$.
Step 8 **If** ($\delta_3 <$ TOL/2) **then**
Output "$x_1, x_2, \ldots, x_{C-1}, \beta_1$", "No more improvement!";
Stop.
Step 9 Set $\delta_2 = \delta_3/2$; $\beta_2 = \beta_{S1}(\mathbf{x} - \delta_2\mathbf{z})$.
Step 10 Set $\nu_1 = (\beta_2 - \beta_1)/\delta_2$; $\nu_2 = (\beta_3 - \beta_2)/(\delta_3 - \delta_2)$; $\nu_3 = (\nu_2 - \nu_1)/\delta_3$.
//Note that we use Newton's forward divided-difference formula [11] to find a quadratic
//polynomial $V(\delta) = \beta_1 + \nu_1\delta + \nu_3\delta(\delta - \delta_2)$ which interpolates $\nu(\delta)$ at $\delta = 0$, $\delta = \delta_2$, $\delta = \delta_3$.
Step 11 Set $\delta_0 = 0.5(\delta_2 - \nu_1/\nu_3)$; $\beta_0 = \beta_{S1}(\mathbf{x} - \delta_0\mathbf{z})$.
//Note that the critical point of V occurs at δ_0.
Step 12 Find δ from $\{\delta_0, \delta_3\}$ so that $\beta = \beta_{S1}(\mathbf{x} - \delta\mathbf{z}) = \min\{\beta_0, \beta_3\}$.
Step 13 Set $\mathbf{x} = \mathbf{x} - \delta\mathbf{z}$.
Step 14 **If** ($|\beta - \beta_1| <$ TOL) **then**
Set $n_1^{\dagger} = \text{Round}(x_1), n_2^{\dagger} = \text{Round}(x_2), \ldots, n_{C-1}^{\dagger} = \text{Round}(x_{C-1})$;
Output "$n_1^{\dagger}, n_2^{\dagger}, \ldots, n_{C-1}^{\dagger}, \beta_{S1}(\mathbf{n}^{\dagger}), k$", and "Success!";
Stop.
Step 15 Set $k = k + 1$.
Step 16 Output "$x_1, x_2, \ldots, x_{C-1}, \beta$", "Maximum iterations exceeded!".
Stop.

Algorithm 1: The algorithm of GRA.

$Q(i)$ according to p_i and λ_i, respectively. Then we also construct the dummy objective function $\beta_{S1}(\mathbf{x})$, and we set the initial value of $\mathbf{x} = [x_1^{(0)}, x_2^{(0)}, \ldots, x_{C-1}^{(0)}]^t$. All these $C - 1$ elements of $\mathbf{x}$ need to satisfy that $\Delta H_i^*/w_i = \Delta H_{i+1}^*/w_{i+1}$ (see Lemma 11). In Steps 3 and 4, we evaluate the magnitude of β_{S1} at $\mathbf{x}$ and determine the steepest decent direction. If a zero gradient occurs, the algorithm will stop. In Steps 5–8, we need to find a new position whose magnitude of β_{S1} is smaller than the current one. Then we move a distance of δ towards the steepest decent direction, and $\mathbf{x}$ is replaced with the new position (Steps 9–13). We check if any stopping criterion is met in Step 14. In Step 15, we employ $K_{\max}$ to limit the total iterations; thus, the algorithm will stop anyway.

5. Comparison with Other Research

In the real world, such minimization problems usually call for instant and near-optimal solutions, especially for large problem instances. As a result, many metaheuristic algorithms have been proposed for providing instant and near-optimal solutions, for example, [22–24]. Although these algorithms achieved better execution time, their solution quality cannot be ensured or bounded.

Therefore, we need deterministic algorithms that are able to converge linearly and achieve near-optimality. Jea et al. [20] solved a basic similar problem, DAP, that is, $\min\sum_{i=1}^{m}[(\sum_{d_j \in M_i} 1)(\sum_{d_j \in M_i} p_j)]$. Here, the number of machines or channels is denoted by m, literally meaning C in this study. The terms in the square bracket are in a very simple form. Even so, the partition problem is also time-consuming for obtaining an optimal solution when the problem size is larger than 50. This problem was solved near-optimally by their proposed algorithm [20]. Wang and Jea [21] proposed another partition problem, SPP, that is, $\min\sum_{i=1}^{m}[(\sum_{d_j \in M_i} c_j)(\sum_{d_j \in M_i} p_j)]$. The terms in the square bracket become slightly complicated. This problem was also solved near-optimally in linear time. Jea and Wang [2] introduced another partition problem, DBAP; that is, $\min\sum_{i=1}^{m}[a_i(\sum_{d_j \in M_i} b_j)(\sum_{d_j \in M_i} 1/n)]$. The terms in the square bracket look similar to those of SPP. However, the transformation between $\mathbf{Z}^n$ and $\mathbf{R}^n$ becomes more difficult. After establishing a complete theoretical basis, this problem was also solved near-optimally.

Table 1: System settings in the experiments.

Parameter	Default	Range	Meaning
N		$1, 2, \ldots, 1000$	Number of data items
C		$1, 2, \ldots, 50$	Number of channels
p_j		Zipf(θ)	Access probability of data item j
λ_j		$N(200\mu, 50\sigma)$	Size of data item j
w_i		$(1 - 0.25R, 1 + 0.25R)$	Bandwidth of channel i
R		0.0, 0.25, 0.5, 0.75	Parameter for bandwidth
θ		0.0, 0.25, 0.5, 0.75	Parameter for access probability
μ		0.0, 0.25, 0.5, 0.75	Parameter for item size
σ		0.0, 0.25, 0.5, 0.75	Parameter for item size
r_c	0.8		Crossover rate for GA
r_m	0.05		Mutation rate for GA
L	100		Population size for GA
$K_{\max}$	10000		Maximum iteration number for GRA
TOL	0.01		Tolerance for GRA

On the other hand, Wang and Chen [25] proposed another minimization problem, $\min\sum_{i=1}^{m} |(\sum_{d_j \in M_i} q_j) - r|$ subject to $\sum_{i=1}^{m} [(\sum_{d_j \in M_i} q_j)(\sum_{d_j \in M_i} p_j)] \le T$ and $\sum_{d_j \in M_i} 1 \le v$ for all nonempty M_i. The constraints and absolute values make the problem harder. It is interesting that the discretized problem could also be solved by the same technique in $\mathbf{R}^n$.

In this study, we propose the WTM problem, that is, $\min\sum_{i=1}^{m} [(\sum_{d_j \in M_i} \lambda_j)(\sum_{d_j \in M_i} p_j)/w_i]$. As shown in Figure 1, WTM is the most complicated form. WTM relates to not only partition but also permutation. In the three problems, DAP, SPP, and DBAP, the position orders of items in $\mathbf{n}^*$ and $\mathbf{x}^*$ are the same. However, for WTM, the position orders of items in $\mathbf{n}^*$ and $\mathbf{x}^*$ might be different. This makes transformation between $\mathbf{Z}^n$ and $\mathbf{R}^n$ more difficult, so we sacrifice some accuracy of position order and force the transformation to be done. Even so, WTM describes a general form of such problems and still achieves near optimality.

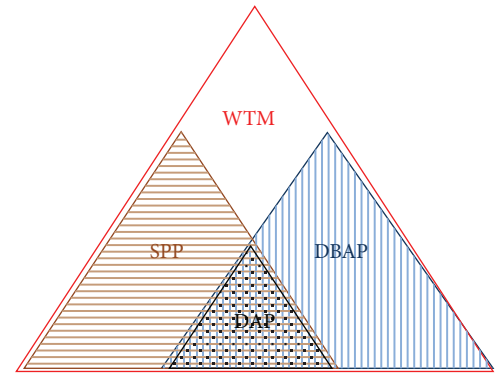


Figure 1: The relationship diagram.

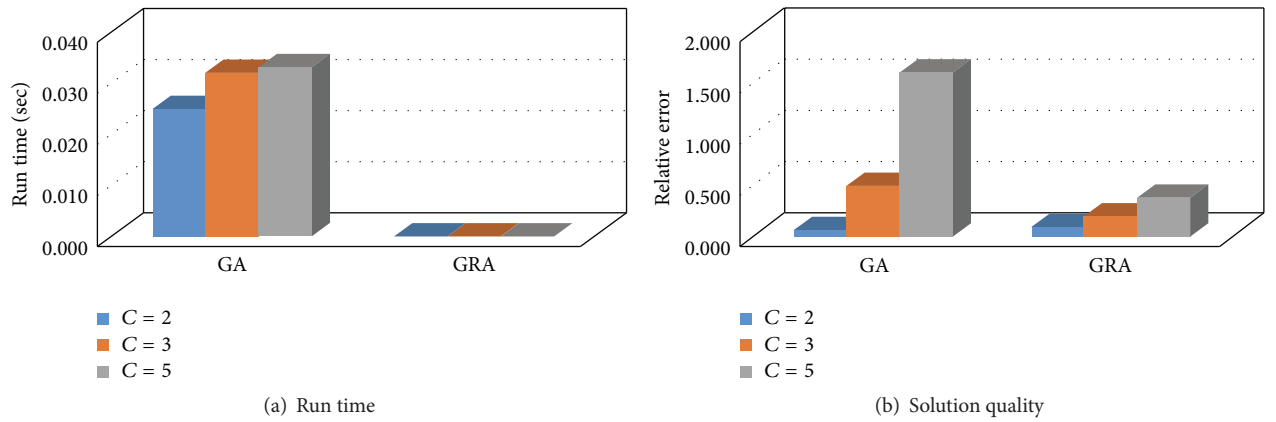
6. Computational Experiments

The experiments are divided into three parts. First, since the real-world situations are complicated, we need to determine some significant system settings. We develop a basic genetic algorithm (GA) to conduct a pilot experiment for determining these settings. Second, when the problem size is small, both algorithms (i.e., GRA and GA) are compared with an exhaustive search algorithm. Third, when the problem size is large, we compare GA and GRA to evaluate their solution quality and execution speed.

Table 1 summarizes the parameters used in the experiments. Parameters N, C, p_j, λ_j, and w_i have already been defined in Section 2. Access probability p_j follows a Zipf distribution with parameter θ [20]. Item size λ_j follows a discrete normal distribution with parameters 200μ and 50σ. Bandwidth w_i follows a discrete uniform distribution DU$(1 - 0.25R, 1 + 0.25R)$. For GA, the population size, crossover rate, and mutation rate are 100, 0.8, and 0.05. For GRA, the maximum iteration number and tolerance are 10,000 and 0.01, respectively. All the proposed algorithms were implemented in PASCAL and executed in a Windows 7 environment on an Intel Xeon E3 1230 @ 3.20 GHz with 8 GB RAM. For each setting, 30 random trials were conducted and recorded.

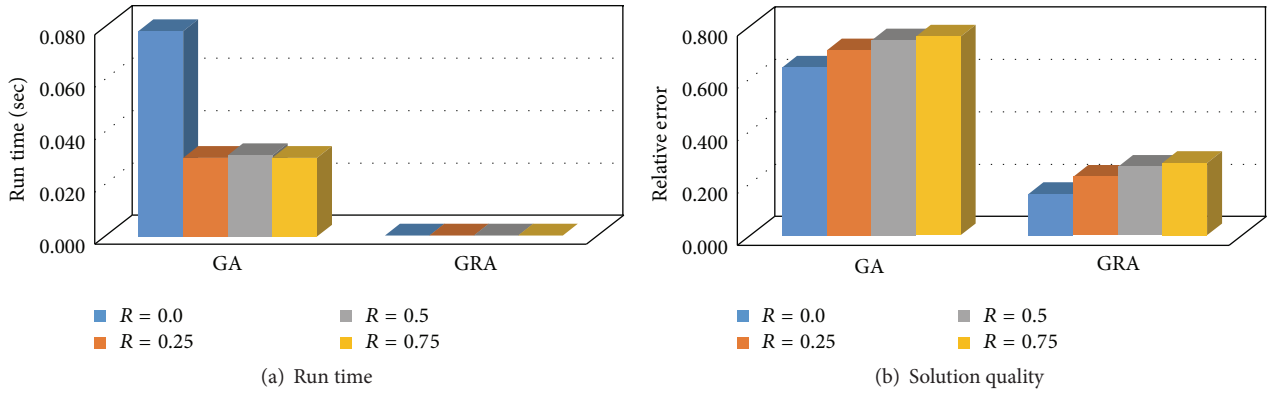
In the first part, we develop a basic genetic algorithm (GA) to test the performance. Each chromosome is randomly generated. For example, let $N = 4$, $C = 2$ and generate $5(= N + C - 1)$ random values, 0.42, 0.95, 0.13, 0.21, and 0.36. According to their magnitudes, the largest number 0.95 means a channel separator. That is, item 4 (having the fourth smallest value, 0.42) is allocated to channel 1, and items 1, 2, and 3 are allocated to channel 2. For each population, there are 100 random chromosomes. The fitness is defined as $f_i = c_i^{-0.5} / \sum_{k=1}^{L} c_k^{-0.5}$, for $i = 1, 2, \ldots, L$, where c_i is the objective cost achieved by the ith chromosome. A standard roulette wheel selection is employed, and two parent chromosomes are selected for generating two child chromosomes by a two-point crossover. Moreover, a simple single-point mutation is adopted. GA will terminate if the run time is over $100n$ milliseconds or no improvement is made during the recent 100 generations.

In the second part, Figure 2 shows the effect of C on the performance for $N = 10$. The relative error is defined



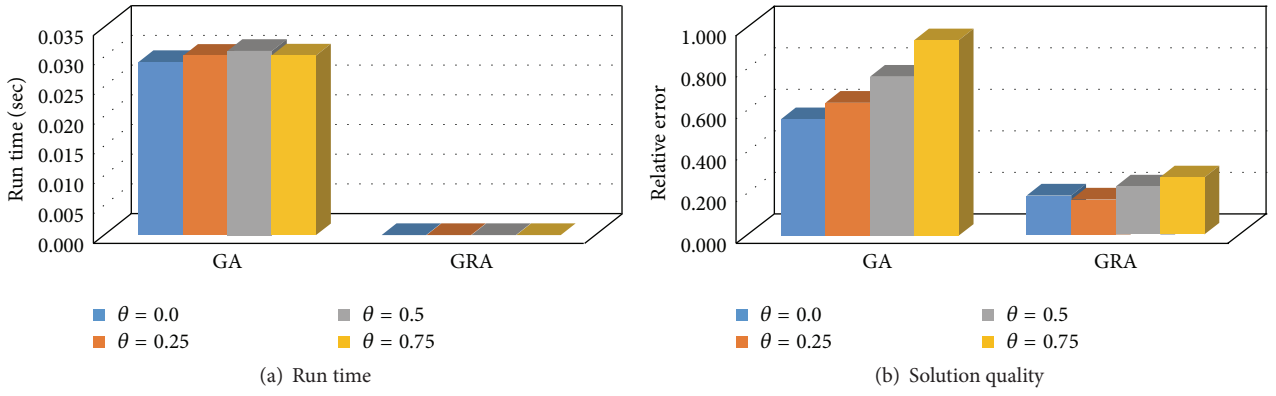
(a) Run time (b) Solution quality

FIGURE 2: The effect of C on performance ($N = 10$).



(a) Run time (b) Solution quality

FIGURE 3: The effect of R on performance ($N = 10$).

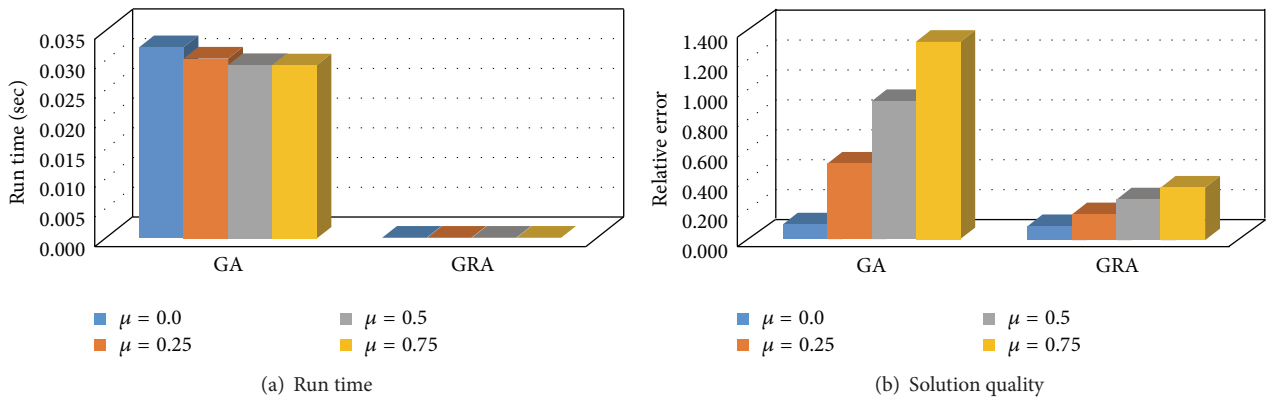


(a) Run time (b) Solution quality

FIGURE 4: The effect of θ on performance ($N = 10$).

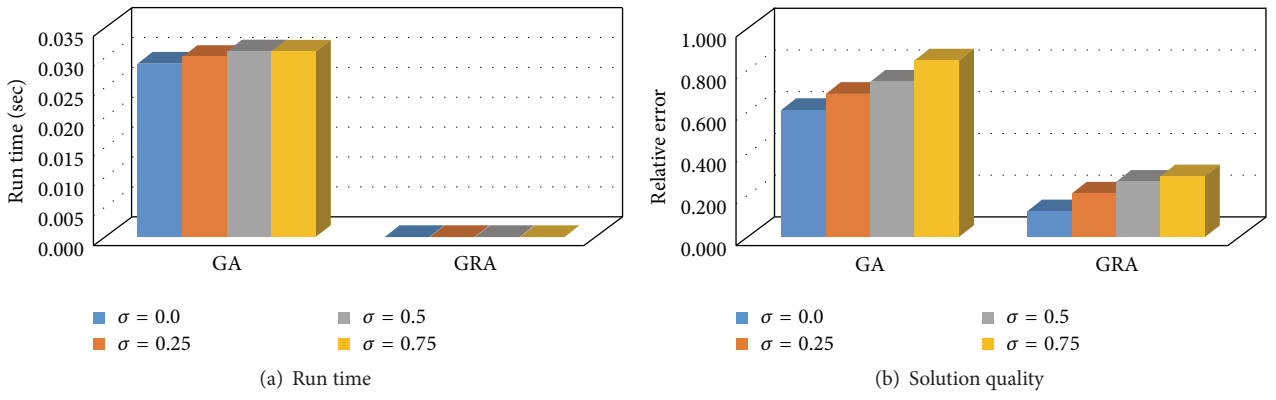
by $(c_{GA} - c_{OPT})/N$ or $(c_{GRA} - c_{OPT})/N$, where c_{GA}, c_{GRA}, and c_{OPT} are objective costs obtained by GA, GRA, and an exhaustive search algorithm. It is seen that GA takes more run time when C increases. Moreover, C does not influence GRA and most instances can be solved within 1 millisecond. On the other hand, GA can easily become trapped in local minimums, since there are many local minimums for such an optimization problem. Consequently, the relative error of GA is relatively higher. Since the setting of $C = 2$ has the least significance, we omit it in later experiments.

Similarly, Figures 3–6 show various effects on the performance for $N = 10$. As shown in Figures 3 and 4, identical access probability ($\theta = 0$) and equal bandwidth ($R = 0$) make the problem easier, so we only observe larger θ and R in later experiments. In Figures 5 and 6, when μ and σ increase, the relative errors increase slightly. The worst mean relative error is 0.353 when $\mu = 0.75$; that is, the data items are large. Therefore, we only test the instances with larger item sizes in later experiments.



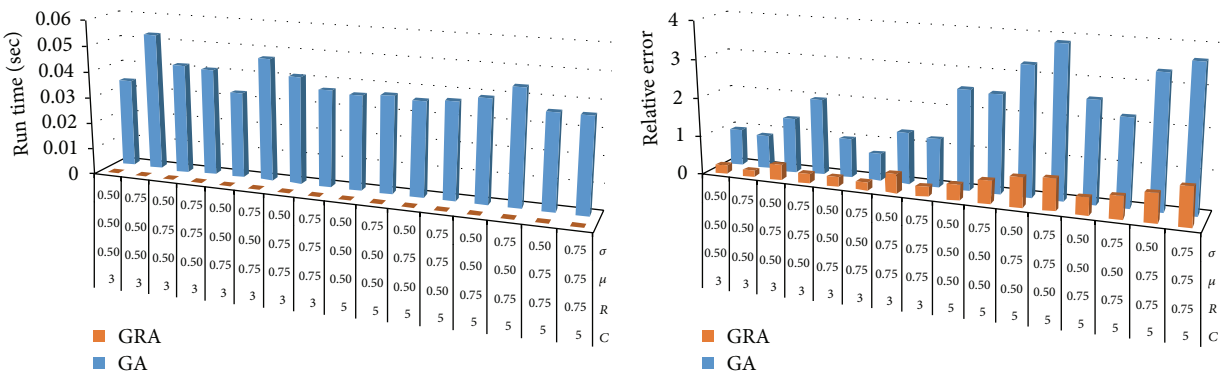
(a) Run time (b) Solution quality

FIGURE 5: The effect of μ on performance ($N = 10$).



(a) Run time (b) Solution quality

FIGURE 6: The effect of σ on performance ($N = 10$).

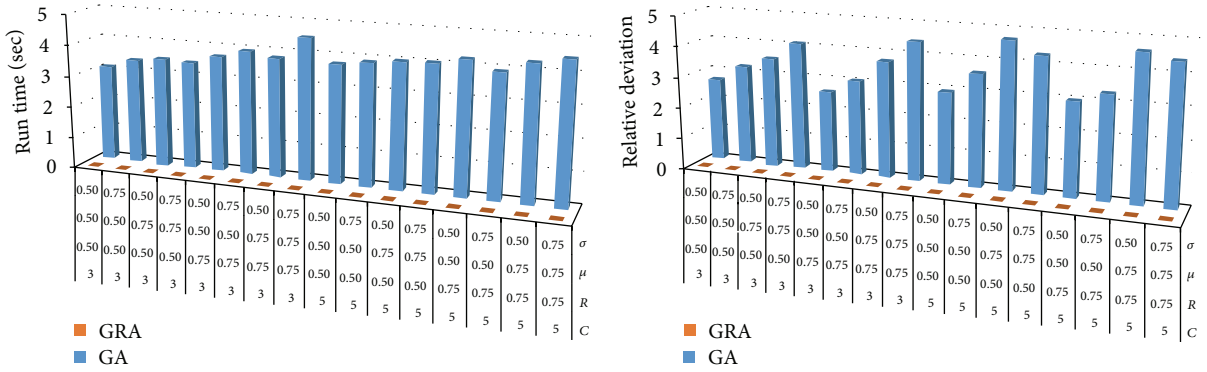


(a) Run time (b) Solution quality

FIGURE 7: Comparison of two algorithms ($N = 12, \theta = 0.5$).

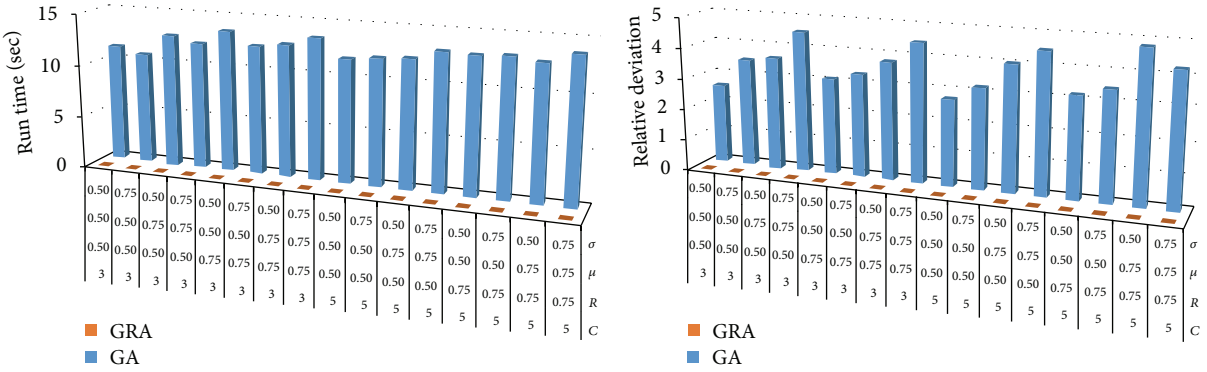
Figure 7 compares two algorithms for $N = 12$ in terms of execution speed and solution quality. Again, GRA almost takes no time to obtain near-optimal solutions, whereas GA needs 3 seconds on average. For the setting of $N = 12$, there are more local minimums than $N = 10$. It becomes more difficult to locate the optimal solution. Therefore, GA takes more run time but still has larger relative errors.

In the third part, Figures 8 and 9 show the performance of two algorithms when the problem size is large. Since we cannot obtain optimal solutions when n is large, we compute their relative deviations. The relative deviations are defined by $(c_{GA} - c_{min})/N$ and $(c_{GRA} - c_{min})/N$, where $c_{min} = \min\{c_{GA}, c_{GRA}\}$. GRA outperforms GA greatly in terms of solution quality and execution speed. For most instances, they can be solved within 1 millisecond for $C = 5$ and $N = 500$. Even for the worst case, GRA takes only 16 milliseconds. On the other hand, GA cannot jump out of local minimums, no matter how many trials it tries. Its solution quality depends



(a) Run time (b) Solution quality

Figure 8: Comparison of two algorithms ($N = 250$, $\theta = 0.5$).



(a) Run time (b) Solution quality

Figure 9: Comparison of two algorithms ($N = 500$, $\theta = 0.5$).

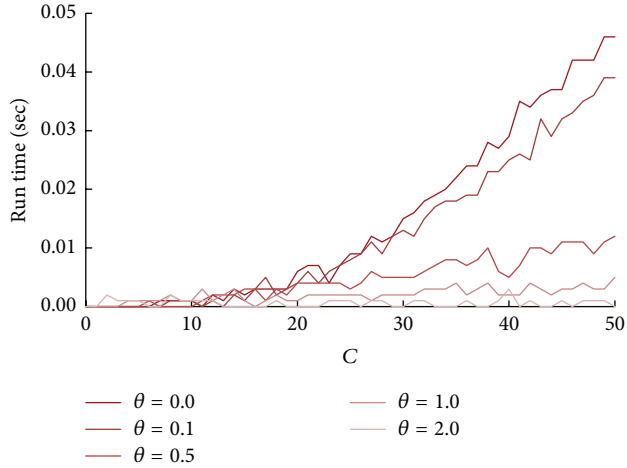


Figure 10: The effect of C on the execution time of GRA.

highly on the initial population. If there are no high-quality solutions at the beginning, it is difficult for GA to locate the optimal solution in the n-dimensional solution space.

Figure 10 shows the effect of C on execution time. To observe the average execution time of GRA, we set $N = 1{,}000$, $\theta = 0.0, 0.1, 0.5, 1.0, 2.0$, $R = 0.5$, $\mu = 0.5$, $\sigma = 0.5$, and $C = 1$ to 50. Since GA cannot compete with GRA for solution quality when $N = 500$, we do not examine the behavior of GA for such a large problem size. As shown in the figure, when C increases, the run time of GRA slightly increases. It means the number of channels is directly proportional to the run time of GRA. In fact, the number of channels of a mobile

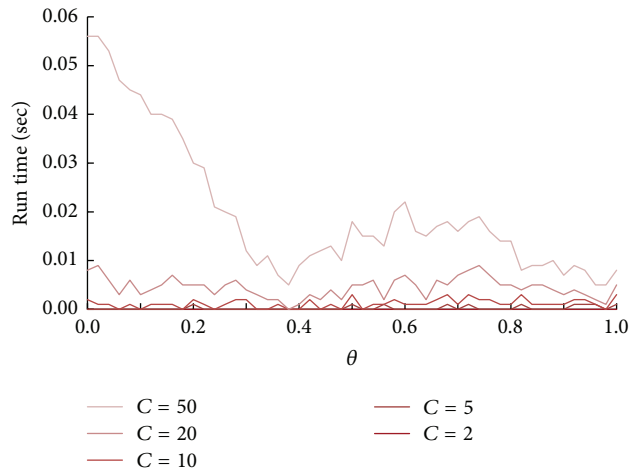
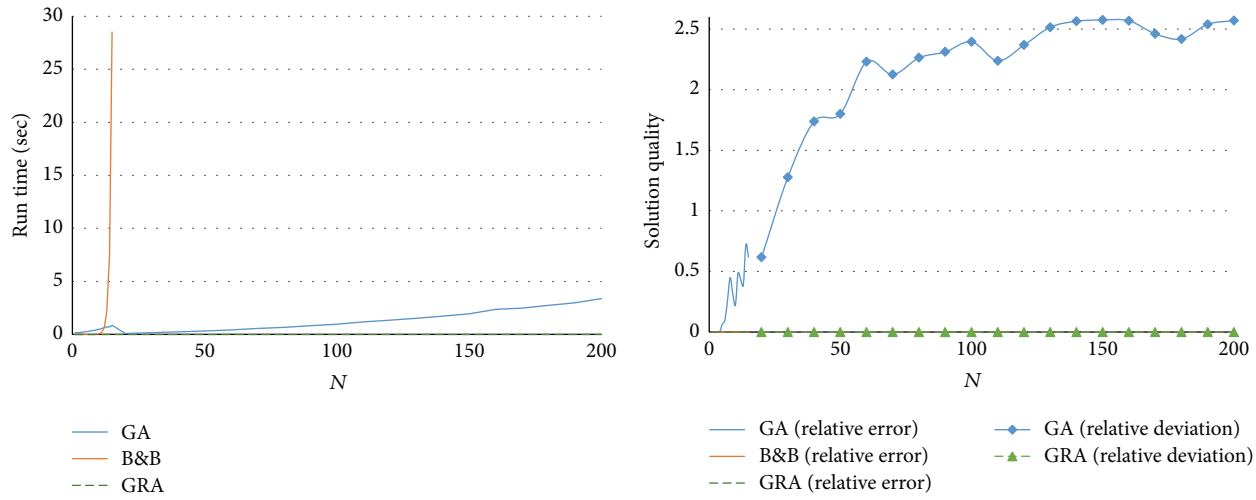


Figure 11: The effect of θ on the execution time of GRA.



(a) Run time

(b) Solution quality

Figure 12: Comparison of three algorithms (m= 3, θ = 0.5, μ = 0.5, and σ = 0.5).

environment is far lower than 50. It implies that GRA is able to deal with scheduling 1000 data items for any real-world broadcast server.

Figure 11 shows the effect of θ on convergence speed. In this experiment, we set $N = 1000$, $C = 2, 5, 10, 20, 50$, $R = 0.5$, $\mu = 0.5$, $\sigma = 0.5$, and $\theta = 0.0$ to 1.0. When $\theta = 0.0$ and $C = 50$, all the data items are of the same popularity. However, the sizes of the data items are different and the bandwidths of the channels are also distinct. These variations make it difficult for GRA to converge. On the other hand, as $\theta = 1.0$, there are only a few popular data items which should be allocated to channel 1 (i.e., that with the highest bandwidth). The other items with very low access probabilities can be roughly allocated to the remaining channels. Therefore, GRA converges very rapidly when $\theta > 0.8$.

In Figure 12, we implement an optimization algorithm, that is, a branch-and-bound algorithm (B&B), and compare it with GA and GRA. Since B&B is very time-consuming, we only observe the results for $N \leq 15$. Both B&B and GRA can provide optimal solutions when $N \leq 15$. On the other hand, the average run time of B&B for $N = 15$ is 30 seconds, whereas the run time of GRA is always less than 4 seconds, even for $N = 200$. It is clear that the time complexity will exclude such optimization algorithms from practical use.

In sum, GRA is a practical algorithm, even for $N = 500$. As compared with the other two algorithms, GRA is more suitable for application in the real world. Moreover, we also guarantee that each solution is generated within a linear time and give an error bound.

7. Conclusion

Minimization problems are usually time-consuming, especially for large problem instances. Consequently, most traditional studies have employed metaheuristic algorithms to solve such problems. However, such algorithms have several shortcomings. First, their solutions are obtained by trial and error, so solution quality is not guaranteed. Second, their approximation algorithms cannot converge linearly. Third, some traditional methods need to keep track of partial results,

so they are memory-consuming. Fourth, some traditional methods, such as dynamic programming and branch-and-bound algorithms, are not scalable. Once the problem size increases, it may take several days to generate an optimal solution, and such a delay is impractical.

Mapping a discretized problem from $\mathbf{Z}^n$ to $\mathbf{R}^n$ is an interesting idea. In this study, a gradient-based algorithm is proposed to deal with WTM. We first map it from $\mathbf{Z}^n$ to $\mathbf{R}^n$. Then the mapped problem is solved optimally in $\mathbf{R}^n$. Finally, the optimal solution is mapped from $\mathbf{R}^n$ back to $\mathbf{Z}^n$. Moreover, the theoretical basis ensures that the proposed algorithm can converge linearly, provide high-quality solutions, and require less memory.

In the near future, we will extend the concept to other optimization problems. By mapping a problem from its original domain to another domain, we are likely to find a more time-efficient and cost-effective way to achieve similar results.

Parameters

N: Number of data items
Λ: Summation of all λ_i
C: Number of channels
S: Access pattern (sequence of (p_i, λ_i))
$S0$: Access pattern for WTM
$S1$: Access pattern for WTM$'$
$P(j)$: Cumulative function defined by p_j $(P(N) = 1)$
$Q(j)$: Cumulative function defined by λ_j $(Q(N) = \Lambda)$
$\mathbf{n}$: Position vector $[n_1, n_2, \ldots, n_{C-1}]^t \in \{1, 2, \ldots, N\}^{C-1}$
$\mathbf{x}$: Position vector $[x_1, x_2, \ldots, x_{C-1}]^t \in (1, N)^{C-1}$
$\alpha_S(\mathbf{n})$: Objective function of WTM
$\beta_S(\mathbf{x})$: Objective function of WTM$'$
$F(x)$: Interpolating function passing through all points $(j, P(j))$
$H(x)$: Interpolating function passing through all points $(j, Q(j))$
$\mathbf{n}^*$: Optimal solution to $\alpha_{S0}(\mathbf{n})$
$\mathbf{n}^\#$: Optimal solution to $\alpha_{S1}(\mathbf{n})$
$\mathbf{x}^*$: Optimal solution to $\beta_{S1}(\mathbf{x})$
$\mathbf{n}^\dagger$: Near-optimal solution to $\alpha_{S0}(\mathbf{n})$.

Competing Interests

The author declares no competing interests regarding the publication of this paper.

References

[1] S. Tainsky and M. Jasielec, "Television viewership of out-of-market games in league markets: traditional demand shifters and local team influence," *Journal of Sport Management*, vol. 28, no. 1, pp. 94–108, 2014.

[2] K.-F. Jea and J.-Y. Wang, "Probabilistic service partition for parallel and distributed computing," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp. 3887–3909, 2010.

[3] S. Y. Chien and Y. T. Lin, "The effects of the service environment on perceived waiting time and emotions," *Human Factors and Ergonomics in Manufacturing & Service Industries*, vol. 25, no. 3, pp. 319–328, 2015.

[4] Z. Jin-Hong, J. Rui-Xuan, and Z. Gui, "Multi-item distribution policies with supply hub and lateral transshipment," *Mathematical Problems in Engineering*, vol. 2015, Article ID 702482, 12 pages, 2015.

[5] Y.-T. Lin, K.-N. Xia, and L.-T. Bei, "Customer's perceived value of waiting time for service events," *Journal of Consumer Behaviour*, vol. 14, no. 1, pp. 28–40, 2015.

[6] B. Zheng, X. Wu, X. Jin, and D. L. Lee, "Broadcast scheduling: TOSA: a near-optimal scheduling algorithm for multi-channel data broadcast," in *Proceedings of the 6th International Conference on Mobile Data Management (MDM '05)*, pp. 29–37, Ayia Napa, Cyprus, May 2005.

[7] 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*, vol. 54, no. 5, pp. 558–572, 2005.

[8] J.-L. Huang and M.-S. Chen, "Dependent data broadcasting for unordered queries in a multiple channel mobile environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1143–1156, 2004.

[9] H.-F. Yu, "Efficient periodic broadcasting scheme for video delivery over a single channel," *Multimedia Tools and Applications*, vol. 74, no. 15, pp. 5811–5824, 2015.

[10] D.-J. Chiang, C.-S. Wang, C.-L. Chen, and D.-J. Deng, "Real-time data delivery using prediction mechanism in mobile environments," *Wireless Personal Communications*, vol. 74, no. 4, pp. 1345–1362, 2014.

[11] R. L. Burden and J. D. Faires, *Numerical Analysis*, Thomson Learning, 9th edition, 2011.

[12] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine, "Efficient data allocation over multiple channels at broadcast servers," *IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1231–1236, 2002.

[13] H. M. Soroush, "Scheduling with job-dependent past-sequence-dependent setup times and job-dependent position-based learning effects on a single processor," *European Journal of Industrial Engineering*, vol. 9, no. 3, pp. 277–307, 2015.

[14] G.-J. Sheen and L.-W. Liao, "A branch and bound algorithm for the one-machine scheduling problem with minimum and maximum time lags," *European Journal of Operational Research*, vol. 181, no. 1, pp. 102–116, 2007.

[15] W.-C. Peng and M.-S. Chen, "Efficient channel allocation tree generation for data broadcasting in a mobile computing environment," *Wireless Networks*, vol. 9, no. 2, pp. 117–129, 2003.

[16] W.-C. Peng, J.-L. Huang, and M.-S. Chen, "Dynamic leveling: adaptive data broadcasting in a mobile computing environment," *Mobile Networks and Applications*, vol. 8, no. 4, pp. 355–364, 2003.

[17] R. C. T. Lee, R. C. Chang, S. S. Tseng, and Y. T. Tsai, *Introduction to the Design and Analysis of Algorithms*, Unalis Press, 1999.

[18] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, Pa, USA, 2000.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, London, UK, 13rd edition, 2009.

[20] K.-F. Jea, J.-Y. Wang, and S.-Y. Chen, "A linearly convergent method for broadcast data allocation," *Computers & Mathematics with Applications*, vol. 56, no. 2, pp. 324–338, 2008.

[21] J.-Y. Wang and K.-F. Jea, "A near-optimal database allocation for reducing the average waiting time in the grid computing environment," *Information Sciences*, vol. 179, no. 21, pp. 3772–3790, 2009.

[22] T. Dede, S. Bekirolu, and Y. Ayvaz, "Weight minimization of trusses with genetic algorithm," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2565–2575, 2011.

[23] A. Karimi, H. Nobahari, and P. Siarry, "Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions," *Computational Optimization and Applications*, vol. 45, no. 3, pp. 639–691, 2010.

[24] C. R. Scrich, V. A. Armentano, and M. Laguna, "Tardiness minimization in a flexible job shop: a tabu search approach," *Journal of Intelligent Manufacturing*, vol. 15, no. 1, pp. 103–115, 2004.

[25] J.-Y. Wang and J.-S. Chen, "A data partition method for mems-based storage devices in a distributed computing environment," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 1, pp. 101–115, 2013.