Information Systems 38 (2013) 82–96

Contents lists available at SciVerse ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Sequence optimization for media objects with due date constraints in multimedia presentations from digital libraries

Feng-Cheng Lin [a], Jen-Shin Hong [b,*], Bertrand M.T. Lin [c]

[a] Geographic Information Systems Research Center, Feng Chia University, Taichung, Taiwan, ROC
[b] Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan, ROC
[c] Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan, ROC

ARTICLE INFO

Article history:
Received 18 May 2009
Received in revised form
20 October 2010
Accepted 6 May 2012
Recommended by: G. Vossen
Available online 28 May 2012

Keywords:
Multimedia presentation scheduling
Presentation lag
Digital library
Due date constraint
Buffer constraint

ABSTRACT

This study investigates sequence optimization of media objects in a multimedia presentation that is dynamically composed from digital libraries. Each media object can be associated with a due date. The aim is to schedule the media objects in a delay-prone network environment such that the overall presentation lag and the due date penalties of the media objects of presentations can be minimized. We formulate the sequencing problem with buffer constraints in the media player into a flowshop scheduling problem and present a reduction strategy with a branch and bound algorithm to derive optimal sequences. The algorithm can be applied in applications with up to a dozen objects to be scheduled. In addition, we propose a modified NEH-based heuristic algorithm which can provide approximate solutions with an average error rate of less than 4%. The computation-efficient heuristic, when deployed in applications with heavily loaded servers, can obtain near-optimal sequences for problems with more than a dozen objects. The proposed algorithms are embedded into a prototype system for providing digital library services.

1. Introduction

In a multimedia digital library system, a query to the database typically retrieves a number of relevant media objects. In general, while most existing systems provide interfaces for users to view the objects by repetitively "clicking, downloading, and playing" the objects one by one, there is a desire of users for a "TV-like" presentation style where the objects are continuously played. A presentation of this kind sequentially combines and continuously presents the media objects such that the user does not have to repetitively click to retrieve and play the media objects. Applications preferring a TV-like presentation would arise from any system that can combine multiple separate media objects into a continuous presentation. Examples include assembling a TV-like documentary based on queries to a multimedia database, continuously showing media items in an online multimedia album, etc. A continuously played TV-like presentation particularly suits hand-held portable devices such as pocket PCs, organizers, and cell phones. The input interfaces of these portable devices usually are more difficult to operate than a mouse. In particular, traditional repetitive "click-and-play" operations are less effective when users are in motion. In practice, to suit more users with different navigation preferences, the design of the navigation interface should provide both click-and-play and TV-like presentations.

In a typical online multimedia presentation application, the media objects are retrieved from a server through the Internet. The total latency of a presentation is thus an important factor in the overall quality of the

* Corresponding author.
E-mail address: jshong@ncnu.edu.tw (J.-S. Hong).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.05.008

service. To cope with the congestion and delay problems, a commonly used strategy is to "prefetch" the media objects before their presentation due time. In an on-the-fly assembled presentation delivered from digital libraries, there is no pre-defined order for presenting the media objects. Hence, the ordering of the objects is determined dynamically on-the-fly. In a typical multimedia presentation, the modalities of the media objects might include text, images, audio, video, vector graphics, etc. Each media type has its unique data compression capabilities. For the same amount of data transmitted, the expected presentation durations for different media types differ drastically. To reduce the total presentation lag, the delivery and presentation order of the media objects should be properly planned.

Media streaming technology is an example application of prefetch technology that aims to reduce the presentation lag. A streaming-based media player prefetches and buffers small chunks of a media object so that the data can be processed before it is rendered. As soon as the buffer is full, the media object starts to play. While the media object is playing, more data is being downloaded. Typically, for a multimedia presentation delivered through a network environment with a reasonable bandwidth, the only playback idle time is the initial download duration for the first chunk, which in general is rather short. Therefore, a practical approach is to present the streamed objects before other non-streamed objects to keep the overall presentation latency low. The main theme of this study is to explore and exploit optimization techniques for ordering the non-streamed media objects of a prefetch-enabled presentation through a slow network such that the presentation lag is minimized.

In general, the end-to-end delay of a presentation depends on: access delay in the server side, communication delay due to network transmission, packetization, buffering, depacketization, and rendering overhead at the player site. In an environment where the bandwidth of the communication channel is rather restricted, the communication delay dominates the end-to-end delay. In this study, we assume that the download time of an object is deterministic. This assumption is reasonable for applications where the end-to-end transmission delay is mainly attributed to a last-mile bottleneck or a server-assigned bandwidth quota limit. In these cases, the end-to-end transmission usually does not exhibit significant bandwidth variations. As compared to commonly-adopted approaches that use random sequences, simulation experiments to be presented in Section 7 nevertheless show that optimized sequences based on the deterministic download transmission time significantly reduce the presentation lags in real life network environments with stochastic variations in the transmission rate.

In Lin et al. [22,23], we provided the essential background and an overview of the prefetch-enabled media object scheduling problems [4] with different real-life constraints. A number of different problem settings have been thoroughly discussed. We identified several problem settings that had never before been explored. In this study, we propose a solution for a specific one of these un-solved problems. Specifically, this study considers the media object scheduling problems with a player-side "buffer constraint" and "due-date" constraints on the media objects. The player-side buffer constraint is common in applications where the multimedia objects are to be presented in small-scale devices, such as pocket PCs and organizers, with restricted memory capacity. The due-date constraints arise from commercial online services in which the media servers are implemented to automatically insert various intermittent media objects for advertisements while the users are viewing a presentation that dynamically combines media objects retrieved from the servers. These advertisement media objects are presented between two neighboring objects in the presentation. Often, an advertisement object is required to be completed within a predetermined time slot in the final presentation, mostly depending on the agreements between service providers and funding supporters. The problems addressed in this paper will be mathematically formulated and computationally solved. Numerical simulations will be described to assess the performance of the proposed algorithms under real-life wireless network conditions.

2. Notations and problem statements

To better illustrate the operations scenarios of the problem setting under study, Fig. 1 shows a Gantt chart illustrating the download and playback of a sequence of media objects. The objective is to find an optimal sequence of media objects with the minimum sum of total presentation time and due date penalties, which are calculated from the overdue times of the media objects. This object sequencing problem could be formulated as a two-machine flowshop scheduling problem subject to a player-side buffer constraint and presentation due date constraints. The correspondence is shown in Table 1.

This study assumes that a media object can be downloaded only if the free space of the buffer is sufficient to accommodate the object. Once the playback of an object is finished, the buffer space occupied by this object is immediately released. In the literature, there are several papers investigating flowshop scheduling with an upper limit on the number of objects that can be allocated in the intermediate storage buffer. For the case without a buffer constraint (denoted as ∞-*buffer* in Papadimitriou and Kanellakis [29]), an optimal solution can be obtained by Johnson's algorithm [16]. For the case in which no job is allowed to be kept in the intermediate storage buffer

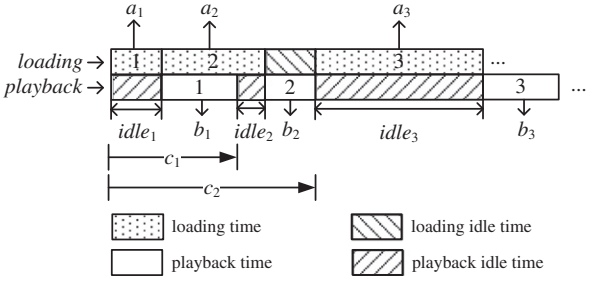

Fig. 1. Illustration of the notations used in problem formulation.

Table 1
Correspondence between object ordering and two-machine flowshop scheduling.

Media object scheduling ordering	Two-machine flowshop scheduling	Notation
A set of media objects	A set of jobs	N
A media object	A job	J_i
Server (sends data to the client)	First machine	M_1
Client (receives/playbacks data from the server)	Second machine	M_2
Download time of a media object	Processing time on first machine	a_i
Playback time of a media object	Processing time on second machine	b_i
Completion time of the playback of a media object	Completion time of job J_i	C_i
Completion time of the presentation	Makespan (the maximum of C_i)	C_{max}

(denoted as *zero-buffer* or *no-wait* in Papadimitriou and Kanellakis [29]), we can use the polynomial algorithm proposed by Gilmore and Gomory [11] for a specific variant of the traveling salesman problem. Beyond these two cases, buffer-constrained flowshop scheduling has been shown to be strongly NP-hard by Papadimitriou and Kanellakis [29,10]. To obtain exact solutions to such a problem with a finite intermediate job-number-based buffer, a number of solution methods have been proposed, for example Leisten [20], Smutnicki [32], Dutta and Cunningham [9], Brucker et al. [5], Tang and Xuan [33], and Wang et al. [35]. In these studies on flowshop scheduling, the number of jobs allowed to keep in the buffer is fixed throughout the process. In the online multimedia applications addressed in this study, we consider the memory size of the player-side buffer instead of the number of objects allowed in the buffer. Therefore, for a given buffer size, the number of objects allowed in the buffer depends on the file sizes of the current playing objects and other unscheduled objects. In an environment with a constant transmission bit rate, the download time of an object (i.e., the processing time on M_1) is proportional to its file size. Jobs of larger sizes require longer processing (or transmission) times. The number of jobs allowed in the buffer thus depends on job processing times. The problem under study is different from conventional buffer-constrained flowshop scheduling. Our buffer model exhibits another unique feature that the buffer resides on machine M_2 rather than acting as an intermediate buffer in between two machines. Therefore, a job currently being processed on machine M_2 resides in the buffer and occupies the buffer space until its completion. In conventional scheduling problems, a job released from the buffer for processing on the next machine immediately frees the space it had acquired. To the best of our knowledge, the scheduling model formulated from the studied object sequencing problem is new to the scheduling community. In the following, we introduce the notation that will be used throughout the paper.

2.1. Notation

$N=\{J_1, J_2, \ldots, J_n\}$ a set of n media objects;
J_i a specific object in N;
a_i download time of media object J_i;
b_i playback time of media object J_i;
M_1 the first machine (server);
M_2 The second machine (client);
C_i completion time of the playback of media object J_i;
$idle_i$ idle time immediately before J_i on machine M_2;
d_i due date of J_i;
T_i tardiness of J_i, i.e. $\max\{C_i - d_i, 0\}$
N_{Exc} a subset of N in which the objects are "exclusive" (i.e., large media objects that cannot reside in the buffer with any others);
N_D a subset of N in which each object is associated with a due date;
N_{Exc-D} $N_{Exc} \cap N_D$ a subset of N_{Exc} in which each object is associated with a due date;
N_A $N_{Exc} \cap \{N \setminus N_D\}$ a subset of N_{Exc} in which no due date is assumed;
N_{BB} $N \setminus N_A$ the object set considered in the branch and bound solution after the problem reduction strategy;
N_{SD} subset of objects of N_{BB} which are already scheduled;
N_{BUF} subset of N_{BB} which are in the buffer;
S_N a particular sequence of jobs of set N_{BB};
$S_{N_{SD}}$ a particular sequence of jobs of set N_{SD};
C_{SD} maximum completion time of the objects in sequence $S_{N_{SD}}$;
C_{max} maximum completion time of the objects in sequence S_N;
T_{max} maximum tardiness in sequence S_N;
ΣT_i sum of tardiness in sequence S_N.

2.2. Problem statements

In the problem addressed in this paper, the optimization goal is to minimize the weighted sum of the makespan (C_{max}) and a due-date related criterion (T_{max} or ΣT_i). Tardiness reflects service quality and makespan indicates system throughput as well as service quality. In practice, the optimization metrics could be either one of the following two formulations of a weighted bi-criteria optimization problem:

Case 1. $\alpha C_{max} + (1-\alpha) T_{max}$

Case 2. $\alpha C_{max} + (1-\alpha) \Sigma T_i$

With the processing-time-dependent buffer constraint, the above two bi-criteria combinations have never before been explored. In both settings, the number of objects

allowed to be kept in the buffer depends on the processing times (a_i) of the jobs on machine M_1. Based on the standard three-field notation introduced in Graham et al. [12], we denote the above two problems by $F2|a_i\text{-buffer}|\alpha C_{max}+(1-\alpha)T_{max}$ and $F2|a_i\text{-buffer}|\alpha C_{max}+(1-\alpha)\Sigma T_i$, respectively. For two-machine flowshop scheduling without any buffer constraint, Lenstra et al. [21] proved that $F2||L_{max}$ and $F2||\Sigma T_i$ are strongly NP-hard. With the processing-time-dependent buffer constraints enforced in both cases, it is clear that $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)T_{max}$ and $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)\Sigma T_i$ inherit complicated structures that are computationally challenging as well.

In the literature, Daniel and Chambers [8], Chakravarthy and Rajendran [6], and Allahverdi [2] have proposed branch-and-bound algorithms and heuristic approaches for solving $F2||\alpha C_{max}+(1-\alpha)T_{max}$. To our knowledge, no solution algorithm has been proposed for $F2||\alpha C_{max}+(1-\alpha)\Sigma T_i$. In this paper, we develop a branch-and-bound algorithm for both problems. Section 3 presents a problem reduction algorithm that decomposes the original problem into a set of independent sub-problems that are relatively easy to solve. Several lower and upper bounds for the two problems are then established.

3. A problem reduction algorithm (PRA)

This section presents a problem reduction strategy that is applied to decompose the original problems into several sub-problems. The problem reduction algorithm (PRA) arrives at a solution with less search efforts than required to solve the original problem. First, we introduce the notion of exclusive object. Given a buffer with a limited capacity, when a relatively large object is being played and thereby occupying the lion's share of the buffer space, the transmission of the next incoming object is prohibited due to the space limit. We term such a media object as an *exclusive object*. That is, whenever an exclusive job is under processing on machine M_2, the processing of other objects on machine M_1 is not allowed.

Consider subset N_D, the subset of N containing the objects constrained by due dates and N_{Exc}, the subset of N containing the exclusive objects. Given the buffer size, we first partition the original media set into two disjoint subsets: exclusive objects without due date constraints (denote by $N_A=N_{Exc}\cap\{N\backslash N_D\}$), and the others (denoted by $N_{BB}=N\backslash N_A$). Whenever an object of N_A is under processing on machine M_2, the processing of other objects on machine M_1 is prohibited. Therefore, by intuition all the objects of N_A should follow all the objects of N_{BB}. An optimal sequence of N_{BB} (denoted by S_{BB}) can be obtained using a branch-and-bound algorithm. The main idea of the PRA is to ignore the objects of N_A in the search process. In this case, the original problem can be downsized to a sub-problem that considers only the ordering of the objects of N_{BB}, and the objects of N_A can be positioned after the derived sequence S_{BB} that minimizes the specified objective function. Such a problem reduction approach is particularly useful when most of the media objects are of sizes comparable to that of the buffer in a given setting.

In the following section, we present a branch-and-bound algorithm for the sub-problem on N_{BB} obtained after applying the reduction step to the original media set. The sub-problem considers only the media set N_{BB}. We propose several lower bounds and upper bounds for this sub-problem. Computational experiments will be conducted to examine the effectiveness and efficiency of the bounds.

4. Lower bounds

This section prosents three lower bounds for problems $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)T_{max}$ and $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)\Sigma T_i$. Development of some lower bounds is adapted from previous results of the single-criterion makespan minimization problem studied in Lin et al. [22].

4.1. Lower bound on C_{max} (LB_1)

Referring to Fig. 2, we consider a node that corresponds to a partial schedule of the search tree. Let N_{SD} be the set of jobs already scheduled, $S_{N_{SD}}$ a sequence of the jobs of set N_{SD}, and C_{SD} the makespan of sequence$S_{N_{SD}}$. Let J_p, J_q, J_r be the last three objects in sequence $S_{N_{SD}}$. Consider the instance when an object J_r is being processed on M_2. A reasonable estimate of the minimum remaining processing time at this node is the total processing time a_i required for all unscheduled objects minus the total allowable processing time on M_1 before all the buffered objects finish their playback.

In the case where the last object J_r is exclusive, the only object residing in the buffer is J_r. Therefore, on M_1 the total allowable processing time before all the buffered objects finish their playback is zero (Fig. 2(a)). In the case where the last object J_r is not exclusive, there could be other buffered objects waiting for processing on machine M_2 (e.g., J_p and J_q in Fig. 2(b)). The total allowable processing time on M_1 before all the buffered objects finish their

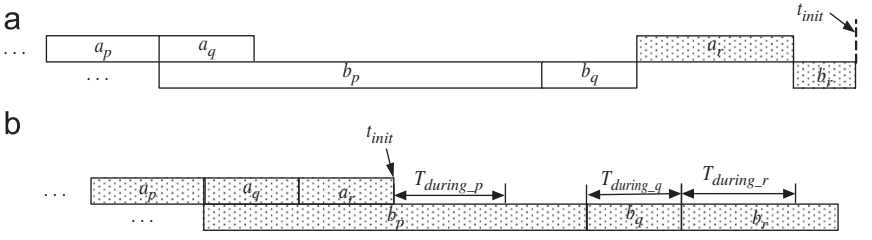

Fig. 2. Two different cases for calculating T_{during_buffer}.

playback is calculated based on the duration that is allowed for further downloading objects during the playback of all buffered objects, denoted by T_{during_buffer} and computed by summing the durations allowed for further download of objects during the processing time of each buffered object J_i (denoted by T_{during_i}). We use the following two values to help compute this value: (1) duration required to fully fill up an empty buffer (denoted by T_{buffer}), and (2) duration allowed to fully fill up the residual free buffer space at the instant the first object in buffer, J_p, starts its playback (denoted by T_{fill}). In the following, we elaborate the derivation of a lower bound using the above terms.

Assume the data transmission bit rate BR on machine M_1 is constant. Then, the time required to completely fill up the empty buffer is $T_{buffer}=BUF/BR$.

For a given partial schedule, the current residual free buffer space BUF_{free} is the buffer size minus the size of the processing object on machine M_2 and other standby jobs in the buffer. Hence, the duration allowed to fill up the residual free buffer space at the instant the first object in buffer starts its playback is $T_{fill}=BUF_{free}/BR$.

There are two different scenarios for calculating the duration allowed for further download of objects during the processing time of an object J_i on M_2 (denoted by T_{during_i}). In the first case (Fig. 3(a)), b_p is larger than T_{fill}. During the processing time of J_p on M_2, the processing on M_1 will be suspended once the buffer is fully filled. The maximum allowable download time is equal to T_{fill}. Hence, $T_{during\_p}=T_{fill}$.

In the second case (Fig. 3(b)), the buffer is not fully occupied before J_q completes on M_2. The buffer has a free space for downloading the next object, say J_r. In such a case, T_{during_q} can be given as b_q.

By combining the above two scenarios, we compute the allowable download time T_{during_i} during the playback of a buffered object J_i as

$$T_{during\_i}=\min(T_{fill},b_i) \tag{1}$$

The total allowable download time for all buffered objects at that node can then be given as

$$T_{during\_buffer}=\sum_{J_i\in N_{BUF}} T_{during\_i} \tag{2}$$

For the case in Fig. 2(a) where the last object belongs to set N_{Exc-D}, it is not possible to download an unscheduled object before the completion of the buffered object. Therefore, $T_{during\_buffer}=0$.

In summary, for a node in the search tree with the last item J_r and sequence $S_{N_{SD}}$, we have the following decision rule to compute the value of T_{during_buffer}:

If $J_r \in N_{Exc-D}$

$T_{during\_buffer}=0$,

Else

$$T_{during\_buffer}=\sum_{J_i\in N_{BUF}} T_{during\_i} \tag{3}$$

We now propose a lower bound based on T_{during_buffer}. Given C_{SD} as the current makespan of the scheduled jobs, the minimum remaining time required to download other unscheduled jobs after the playback of all the currently buffered objects is

$$T_R=\max\left\{\sum_{J_i\in N\setminus N_{SD}} a_i - T_{during\_buffer},0\right\} \tag{4}$$

In the following paragraphs, we present methods that maximize T_{during_p} for each buffered object. Fig. 4 depicts an example where J_p, J_q, and J_r are scheduled and J_i, J_j, and J_k are unscheduled. Given a partial schedule where J_p starts playing, there are two cases to consider for calculating T_{during_p} (duration allowed for further download during the processing time of J_p).

In the first case (Fig. 4(a)), b_q is smaller than T_{fill}. The buffer will not be fully occupied before the completion of J_q. In such a case, $T_{during\_q}=b_q$. In the second case (Fig. 4(b)), b_p is larger than the T_{fill}. The maximum allowable download time T_{during_p} depends on possible combinations of the unscheduled jobs. For example, in Fig. 4(b), a_i and a_k occupy more buffer space than a_j. In addition, a_j and a_k cannot be allocated in the buffer at the

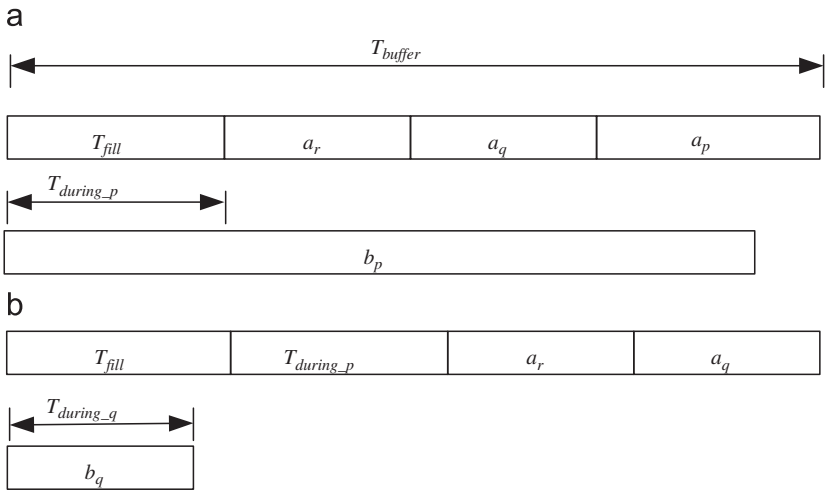

Fig. 3. Two different scenarios for calculating T_{during_i}.

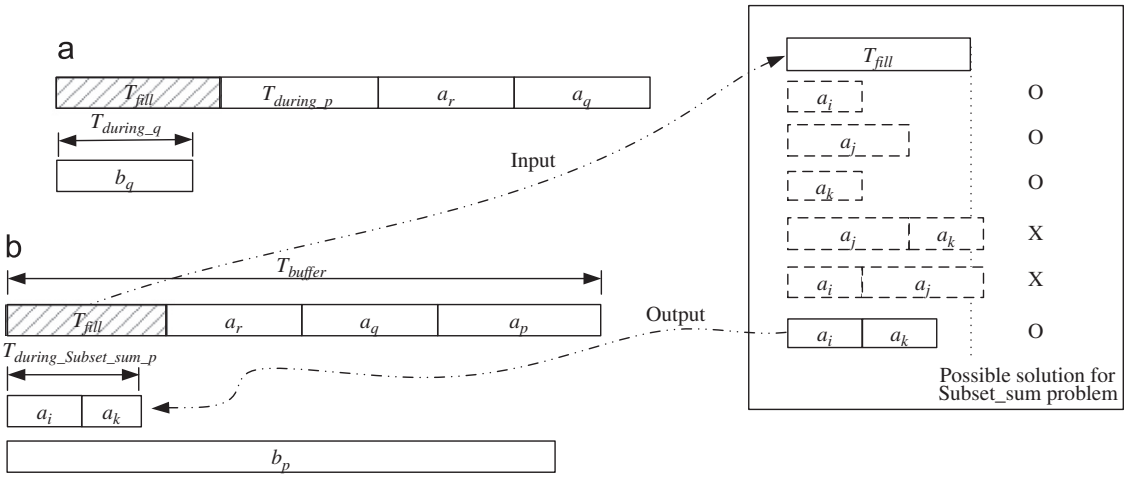

Fig. 4. Calculation of the maximum allowable download time during the playback of an object.

same time since the sum of their download times is larger than T_{fill}. In practice, the optimal combination of the unscheduled jobs that maximizes T_{during_i} (i.e., buffer usage) can be obtained by solving the Subset-sum problem ([24,17]). In the Subset-sum problem, we wish to find a subset of w_1, w_2,…, w_j whose sum is as large as possible but no greater than a given capacity U. Subset-sum is formulated as

$$\text{Maximize} \quad \sum_{j\in N_{BB}\setminus N_{SD}} w_j x_j$$

$$\text{Subject to} \quad \sum_{j\in N_{BB}\setminus N_{SD}} w_j x_j \le U,$$

$$\text{where } x_j \in \{0,1\} \tag{5}$$

To maximize T_{during_p}, we correspond a_i to weight w_i, and T_{fill} to capacity U in Subset-sum. The objective is then to determine an optimal combination of a_i's among the unscheduled jobs for which T_{during_i} is maximized. Subset-sum can be solved by dynamic programming algorithms (DP) in $O(kT_{fill})$ time [17], where k is the number of objects in $N_{BB}\setminus N_{SD}$. We denote the solution obtained via Subset-sum by $T_{during\_Subset\_sum\_p}$ for J_p.

By combining the above two cases, the allowable download time during the playback of a buffer object J_i is given by

$$\text{If } T_{fill} > b_i, T_{during\_i} = b_i \quad \text{else} \quad T_{during\_i} = T_{during\_Subset\_sum\_i} \tag{6}$$

Given $T_R = \max\{\sum_{J_i\in N\setminus N_{SD}} a_i - T_{during\_buffer}, 0\}$, T_R can be maximized by using $T_{during\_i} = T_{during\_Subset\_sum\_i}$ for each buffered object.

We now proceed to estimate the makespan using T_{during_buffer}. In the case where the total processing time on M_1 of all unscheduled objects is not smaller than T_{during_buffer} (i.e., $\sum_{J_i\in N_{BB}\setminus N_{SD}} a_i \ge T_{during\_buffer}$), the makespan should be the sum of the following values: (1) the current makespan C_{SD}, (2) the remaining time required to process all unscheduled objects on M_1 after the playback of all buffered objects (T_R), and (3) the total download idle time (denoted by T_{DI}, the time that cannot be allocated for processing on M_1 when the unscheduled objects are under playback on M_2). A basic lower bound for the makespan can be given by $C_{SD}+T_R+T_{DI}$. In the following, we elaborate the process for deriving T_{DI}.

During the playback of the buffered jobs on M_2 (i.e., within C_{SD}), there are T_{during_buffer} units of time for downloading other unscheduled objects on M_1. The minimum remaining time required to download the unscheduled jobs after the completion time of currently buffered jobs is thus given by $T_R = \sum_{J_i\in N_{BB}\setminus N_{SD}} a_i - T_{during\_buffer}$. Due to the buffer constraint, the time span for downloading all unscheduled jobs could be larger than T_R (Fig. 5(b)). When an object J_i is processed on M_2, the buffer is at least occupied by J_i. Hence, the maximum allowable download time on M_1 during its playback is $T_{buffer}-a_i$. When $T_{during\_i} < b_i$ (Fig. 6(a)), the minimum download idle time $T_{di\_i} = b_i - (T_{buffer}-a_i)$. Otherwise, $T_{di\_i}=0$ (Fig. 6(b) and (c)). Therefore, T_{di_i} can then be given as

$$T_{di\_i} = \max\{b_i - (T_{buffer}-a_i), 0\} \tag{7}$$

Referring to Fig. 7, when $T_{fill} > b_i$ (Fig. 7(a)), $T_{di\_i}=0$. On the other hand, when $T_{fill} \leqq b_i$ (Fig. 7(b)), T_{during_i} can be computed by a dynamic program of the corresponding Subset-sum problem. The download idle time for J_i can then be given as

$$T_{di\_i} = b_i - T_{during\_Subset\_sum\_i} \tag{8}$$

We conclude that T_{di_i} can be given by the following rule:

$$\begin{aligned} &\text{If } J_r \in N_{Exc-D} \\ &\quad T_{di\_i} = b_i, \\ &\text{Else} \\ &\quad \text{If } T_{fill} > b_i, T_{di\_i} = 0, \text{ else } T_{di\_i} = b_i - T_{during\_Subset\_sum\_i} \end{aligned} \tag{9}$$

The minimum download idle time for all unscheduled objects is the sum of the download idle time of all unscheduled objects

$$\begin{aligned} T_{DI} &= \sum\nolimits_{J_i\in\{N_{BB}\setminus N_{SD}\cap N_{BB}\setminus N_{Exc-D}\}} T_{di\_i} \\ &= \sum\nolimits_{J_i\in\{N_{BB}\setminus N_{SD}\cap N_{BB}\setminus N_{Exc-D}\}} \max\{b_i - T_{during\_Sumset\_sum\_i}, 0\} \end{aligned} \tag{10}$$

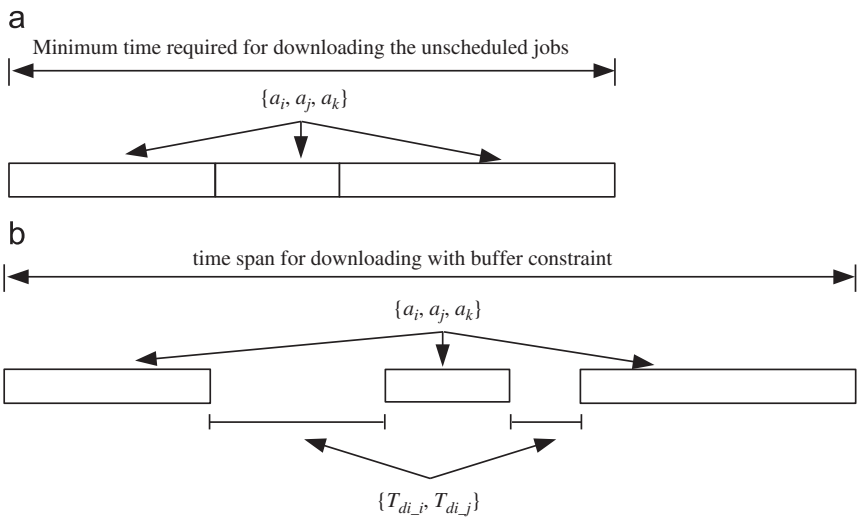

Fig. 5. Calculation of the time required to download all unscheduled jobs after the playback of all buffered objects.

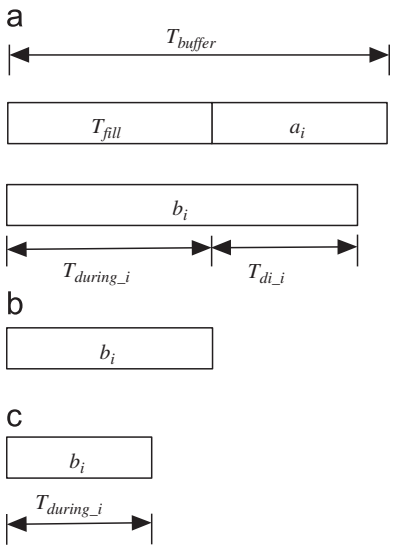

Fig. 6. Calculation of minimum download idle time T_{di_i} of J_i.

In summary, denoting the set of $(N_{BB}\backslash N_{SD})\cap(N_{BB}\backslash N_{Exc-D})$ by N_{set}, we have the following rule for deriving a lower bound on makespan for different situations:

If ($T_{during\_buffer}=0$ or $C_{SD}=0$), then

$$LB_1=\max\left\{\begin{array}{l} C_{SD}+a_{min}+\sum\limits_{J_i\in N_{set}} b_i+\sum\limits_{J_i\in\{N_{BB}\backslash N_{SD}\}\cap N_{Exc-D}}\{a_i+b_i\},\\ C_{SD}+\sum\limits_{J_i\in N_{set}} a_i+T_{DI}+\sum\limits_{J_i\in\{N_{BB}\backslash N_{SD}\}\cap N_{Exc-D}}\{a_i+b_i\}\end{array}\right\}$$

Else

$$LB_1=\left\{\begin{array}{ll} C_{SD}+\sum\limits_{J_i\in N_{set}} b_i+\sum\limits_{J_i\in\{N_{BB}\backslash N_{SD}\}\cap N_{Exc-D}}\{a_i+b_i\} & \text{if } \sum\limits_{J_i\in N_{BB}\backslash N_{SD}} a_i<T_{during\_buffer}\\ C_{SD}+T_R+T_{DI}+\sum\limits_{J_i\in\{N_{BB}\backslash N_{SD}\}\cap N_{Exc-D}}\{a_i+b_i\} & \text{otherwise.}\end{array}\right\} \quad (11)$$

4.2. LB_2: Lower bound on T_{max}

Let C_{SD} denote the completion time of the scheduled jobs, T'_{max} the maximum tardiness among the scheduled jobs, and t_{init} the download initialization time for the next coming job J_i (Fig. 2). A lower bound on the maximum tardiness can be given as

$$LB_{2a}=\max_{i\in N\backslash N_{SD}}\{\max\{\max\{t_{init}+a_i,C_{SD}\}+b_i-d_i,0\},T'_{max}\} \quad (12)$$

In principle, LB_{2a} can be considered as a general lower bound for problems with due date constraints. In the literature, several lower bounds for two-machine flow-shop scheduling with due date related objectives have been proposed, for example, Sen et al. [31], Kim [18], Pan and Fan [26], and Pan et al. [27]. Pan et al. [27] showed that their lower bound outperforms those of Sen et al. [31] and Kim [18]. Therefore, this study deploys a lower bound on T_{max} which is generalized from Pan et al. [27]

$$LB_{2b}=\max_{i\in N\backslash N_{SD}}\{\max\{\max\{t_{init}+P_{i1}+b_k(S_2),C_{SD}+P_{i2}\}-S_{EDD_i},0\},T_{max'}\} \quad (13)$$

where P_{i1} and P_{i2} are the sums of the i shortest processing times among all the jobs on M_1 and M_2, respectively, and S_{EDD_i} is the job in the ith position of an *earliest due date* (*EDD*) sequence of the unscheduled jobs, and $b_k(S_2)$ is the processing time of the job sequenced in the kth position of the SPT schedule on M_2.

With LB_{2a} and LB_{2b}, we readily have a lower bound LB_2 given as follows:

$$LB_2=\max\{LB_{2a},LB_{2b}\} \quad (14)$$

4.3. LB_3: Lower bound on ΣT_i

Similar to the principles described in Section 4.2 for obtaining the lower bounds on T_{max}, given that the current sum of due date penalties is known to be $\Sigma T'$, we can readily compute the lower bounds on ΣT_i as given

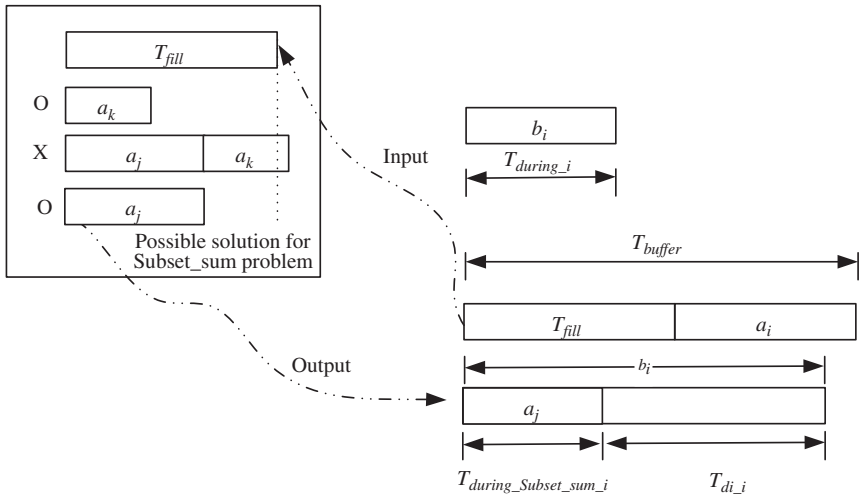

Fig. 7. Calculation of T_{during_i} by solving Subset-sum.

in Eqs. (15) and (16)

$$LB_{3a} = \sum_{i \in N \setminus N_{SD}} \max\{\max\{t_{init} + a_i, C_{SD}\} + b_i - d_i, 0\} + \sum T' \quad (15)$$

$$LB_{3b} = \sum_{i \in N \setminus N_{SD}} \max\{\max\{t_{init} + P_{i1} + b_k(S_2), C_{SD} + P_{i2}\} - S_{EDD_i}, 0\} + \sum T' \quad (16)$$

The two lower bounds together imply the following one

$$LB_3 = \max\{LB_{3a}, LB_{3b}\} \quad (17)$$

4.4. Aggregate lower bounds for the bi-criteria problems

Based on the lower bounds on the different objectives described above, we can readily obtain two lower bounds for the bi-criteria problems using a weighted sum of lower bounds on T_{max} and ΣT_i

$$LB_4 = \alpha LB_1 + (1-\alpha) LB_2 \quad (18)$$

$$LB_5 = \alpha LB_1 + (1-\alpha) LB_3 \quad (19)$$

5. Upper bounds

This section is devoted to the development of three heuristics that are modified from the NEH algorithm [25] to provide approximate solutions or upper bounds. The NEH method has been applied in various makespan minimization problems with impressive performances (for example, [1,30]). Two general dispatching rules, *Earliest Due Date First* (*EDD*) and *Shortest Processing Time First* (*SPT*), are incorporated into our NEH-based heuristic algorithms. The *EDD* rule sequences the jobs in non-decreasing order of their due dates [15] and is commonly used to minimize the maximum lateness among jobs for numerous scheduling problems. The *SPT* rule schedules jobs in ascending order of their processing times and is commonly in the minimization of the average waiting time of jobs.

The basic idea in the NEH approach is to first optimally schedule the first two objects; then place the remaining jobs by decreasing order of $a_i + b_i$, one by one, in one of the positions between the scheduled jobs such that the makespan is minimum at the step. In the following, we present three heuristics, where UB_1 is the NEH heuristic, and UB_2 and UB_3 are modified from the NEH heuristic by deploying the *EDD* and *SPT* rules. We outline the main steps of the modified NEH-based upper bounds (termed as MNEH).

Algorithm UB_1.

Step 1: Sort the objects in non-increasing order of $a_i + b_i$.
Step 2: Take the first two objects in the sorted list, and sequence the two objects to minimize the bi-criteria objective function.
Step 3: Insert the 3rd object into the partial schedule. There are three positions for inserting the 3rd object. Among the three positions, select the position that achieves the smallest bi-criteria objective function value.
Step 4: For $k=4$ to n do
Insert the kth object into the current partial schedule. There are k possible positions for this insertion. Select the position that achieves the smallest the bi-criteria objective function value.
Step 5: Output the final sequence given at the completion of Step 4.

Algorithm UB_2.

Step 1: Apply the *EDD* rule to *ND* (the set of objects with due dates). Apply the SPT rule to the objects of $N \setminus ND$ on machine M_1.
Step 2: Take the first two objects in *ND*, and sequence the two objects to minimize the bi-criteria objective function ($k=2$).

Step 3: Insert the 3rd object in $N\backslash ND$ into the partial schedule if $|ND|>2$. From among the three positions, select the one which achieves the smallest bi-criteria objective function values ($k=3$).
Step 4: For $k=4$ to $|ND|$ do ($|ND|>3$)
Insert the kth object into the previous partial schedule. Select the position that achieves the smallest bi-criteria objective function values.
Step 5: For $p=1$ to $n-|ND|$ do
Insert the pth object into the schedule of $N\backslash ND$ obtained in the previous step. There are $k+p$ possible positions for this insertion. Select the position that achieves the smallest bi-criteria objective function values.
Step 6: Output the final sequence given at the completion of Step 5.

Algorithm UB_3.

Step 1: Apply the *EDD* rule to N_D (the set of objects with due dates). Apply the *SPT* rule to the objects of $N\backslash N_D$ on machine one.
Step 2: Take the elements of N_D as the initial sequence.
Step 3: Insert the first object from $N\backslash N_D$ into the partial schedule. There are $|N_D|+1$ possible positions for inserting the $(|N_D|+1)$th object. Select the position that achieves the smallest bi-criteria objective function values.
Step 4: For $k=2$ to $n-|N_D|$ do
Insert the kth object into the previous partial schedule. There are $|N_D|+k$ possible positions for this insertion. Select the position that achieves the smallest bi-criteria objective function values.
Step 5: Output the final sequence reported at the completion of step 4.

Finally, we select the smallest value among UB_1, UB_2, and UB_3 as the upper bound (UB).

$$UB=\min\{UB_1,UB_2,UB_3\} \tag{20}$$

6. Computational results

This section presents the computational experiments designed and conducted to evaluate the effectiveness and efficiency of the proposed lower bounds and upper bounds. The simulation programs were written in C++ language, and the experiments were performed on IBM × Series × 206 m computers with 3.4 GHz Dual-CPU and 2 GB RAM, running Microsoft Windows Server 2003. All run times reported are in CPU-seconds. Computational experiments were conducted with different problem sizes. Two buffer sizes, 16,000 KB and 30,720 KB, were adopted. We assumed that 20% of the media objects in the original media set are associated with due date constraints. It was given that $Y=\sum_{i=1}^{n} b_i$ as the sum of playback times of all media items, and due dates d_i were randomly drawn from the uniform interval [0, 0.75Y].

For each experiment, 50 independent randomly generated object sets were tested. The processing times a_i and b_i were randomly generated from uniform interval [1,100]. Computational results are summarized in Tables 2–5, where the average number of nodes (*Avg_nodes*), the maximum number of nodes (*Max_node*), and the average computation time (*Avg_t*) for obtaining the optimal solution are presented. The branch-and-bound algorithm was aborted when 600 CPU seconds expired before reporting the optimal solution. Among each 50 data sets, the number of problems that could not be optimally solved was recorded (denoted by "NF" in the tables). We list the results for problems with different weights, α, ranging from 0.1 to 0.9, in the objective functions. Note that for $\alpha=1$, the bi-criteria problem becomes the makespan minimization problem; similarly, $\alpha=0$ dictates the maximum tardiness or total tardiness minimization problem.

The experiments were conducted for four problems sizes with $n=10$, 12, 14, and 16. The computation results for instances of $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)T_{max}$ are given in Tables 2 and 3 for different buffer sizes. From Table 2, it is clear that the proposed branch-and-bound algorithm solved all the problems with $n=10$, 12, or 14 objects, but could fail for $n=16$. Table 3 reveals that as the buffer size decreases, the number of visited nodes and the computation time increase when compared with the results for the large buffer size problems listed in Table 2. In this case, as the value of α increases, the average number of visited nodes and the average computation time increase as well. The computational results for the problems with the objective of $\alpha C_{max}+(1-\alpha)T_{max}$ are given in Tables 4 and 5. The results reveal similar information as those shown in Tables 2 and 3. As a general observation, when α increases (that is, the makespan is more emphasized in the objective function than due date penalties) or buffer size decreases, more computation time is required for finding optimal solutions.

To investigate the performance of the proposed heuristic algorithms, we present in Table 6 the average error ratios between the optimal solutions (*OPT*) and the heuristic solutions (*UB*) for each media set. Error ratio is defined by $(UB-OPT)100\%/OPT$. In all the tested cases, the error ratios between the heuristic solutions and the optimal solutions are less than 4% for problems with smaller buffers and less than 3% for those with larger buffers. The probability that a heuristic produced optimal solutions increases as the buffer size increases. In either case, the results indicate that for most of the test cases the heuristic algorithm constructed approximate solutions that are close to the optimal ones.

7. Prototype system implementation

To demonstrate the practical significance of an optimal schedule in reducing the presentation lag for a presentation delivered through a real world network environment, we have implemented a prototype system and conducted experiments under various settings. Our implementation followed the system architecture depicted in Fig. 8. The following paragraphs discuss the system components in detail:

1) *Multimedia Database.* The database manages the objects and provides mechanisms for retrieving

Table 2
Results for $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)T_{max}$ with a 30,720 KB buffer.

N	10			12			14			16	
α	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	B&B	NF
0.1	4.18E+03	2.39E+04	0.009	1.17E+05	2.41E+06	0.183	1.60E+06	2.10E+07	3.643	NA	6
0.2	3.46E+03	3.63E+04	0.007	7.64E+04	2.41E+06	0.112	6.27E+05	1.07E+07	1.184	NA	6
0.3	7.31E+03	9.52E+04	0.013	1.60E+04	1.87E+05	0.028	3.94E+06	7.89E+07	6.857	NA	4
0.4	3.29E+03	4.21E+04	0.007	7.85E+04	2.41E+06	0.112	1.44E+06	4.25E+07	3.392	NA	4
0.5	1.92E+03	2.31E+04	0.002	1.08E+04	1.72E+05	0.016	1.47E+05	3.18E+06	0.364	NA	2
0.6	5.58E+03	9.19E+04	0.010	1.87E+04	1.95E+05	0.028	2.65E+06	7.89E+07	3.703	NA	4
0.7	3.18E+03	2.45E+04	0.006	4.21E+04	4.79E+05	0.062	6.57E+05	1.34E+07	1.523	NA	8
0.8	4.64E+03	3.87E+04	0.007	8.83E+04	2.41E+06	0.123	6.15E+05	6.80E+06	1.237	NA	5
0.9	1.36E+04	3.80E+05	0.020	1.39E+05	9.34E+05	0.230	2.00E+06	3.20E+07	4.433	NA	3

Table 3
Results for $F2|a_i\text{-}buffer|\alpha C_{max}+(1-\alpha)T_{max}$ with a 16,000 KB buffer.

n	10			12			14			
α	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	NF
0.1	1.02E+04	7.51E+04	0.026	1.19E+05	1.30E+06	0.299	2.05E+06	2.68E+07	5.554	0
0.2	9.58E+03	7.65E+04	0.023	1.27E+05	1.40E+06	0.322	2.76E+06	6.01E+07	7.312	0
0.3	9.86E+03	7.76E+04	0.025	1.28E+05	1.48E+06	0.340	3.09E+06	7.21E+07	8.015	0
0.4	1.14E+04	9.09E+04	0.027	1.40E+05	1.62E+06	0.379	3.51E+06	8.61E+07	9.129	0
0.5	1.11E+04	1.05E+05	0.026	1.50E+05	1.81E+06	0.429	3.92E+06	9.93E+07	10.448	0
0.6	1.26E+04	1.18E+05	0.033	1.82E+05	2.18E+06	0.559	5.28E+06	1.44E+08	14.276	0
0.7	1.55E+04	1.37E+05	0.040	2.61E+05	2.88E+06	0.842	7.35E+06	2.04E+08	21.274	0
0.8	2.15E+04	1.92E+05	0.058	4.73E+05	4.60E+06	1.555	1.13E+07	2.75E+08	36.278	0
0.9	3.56E+04	2.96E+05	0.094	1.56E+06	2.30E+07	4.367		NA		4

Table 4
Results for $F2|a_i\text{-}buffer|\alpha C_{\max}+(1-\alpha)\Sigma T_i$ with a 30,720 KB buffer.

n	10			12			14			16	
α	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	B&B	NF
0.1	3.55E+03	2.27E+04	0.008	1.10E+05	2.41E+06	0.165	2.67E+06	7.21E+07	9.435	NA	6
0.2	3.09E+03	3.39E+04	0.004	7.39E+04	2.41E+06	0.101	7.21E+05	1.13E+07	1.607	NA	5
0.3	6.59E+03	9.19E+04	0.011	1.48E+04	1.56E+05	0.020	2.89E+06	7.89E+07	4.314	NA	5
0.4	3.13E+03	3.94E+04	0.005	7.13E+04	2.41E+06	0.092	7.72E+05	1.14E+07	1.605	NA	4
0.5	1.73E+03	2.31E+04	0.002	8.11E+03	1.70E+05	0.010	1.46E+05	3.18E+06	0.332	NA	2
0.6	5.23E+03	9.19E+04	0.009	3.46E+04	9.07E+05	0.070	2.32E+06	7.89E+07	2.884	NA	5
0.7	2.95E+03	2.45E+04	0.005	3.35E+04	4.79E+05	0.054	5.54E+05	9.07E+06	1.186	NA	7
0.8	3.82E+03	2.61E+04	0.006	8.06E+04	2.41E+06	0.109	7.29E+05	1.28E+07	1.649	NA	4
0.9	9.34E+03	1.99E+05	0.015	7.03E+04	7.11E+05	0.113	1.64E+06	2.89E+07	3.454	NA	4

Table 5
Results for $F2|a_i\text{-}buffer|\alpha C_{\max}+(1-\alpha)\Sigma T_i$ with a 16,000 KB buffer.

n	10			12			14		
α	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t	Avg_nodes	Max_node	Avg_t
0.1	9.54E+03	7.51E+04	0.024	1.09E+05	1.30E+06	0.267	2.53E+06	5.79E+07	5.879
0.2	8.97E+03	7.65E+04	0.022	1.14E+05	1.40E+06	0.279	2.78E+06	6.85E+07	6.435
0.3	9.34E+03	7.68E+04	0.022	1.19E+05	1.48E+06	0.298	3.00E+06	7.58E+07	6.998
0.4	1.06E+04	7.85E+04	0.025	1.30E+05	1.60E+06	0.333	3.81E+06	1.12E+08	8.751
0.5	1.03E+04	8.60E+04	0.025	1.36E+05	1.76E+06	0.366	4.12E+06	1.21E+08	9.682
0.6	1.16E+04	9.84E+04	0.030	1.63E+05	2.08E+06	0.447	4.84E+06	1.40E+08	11.699
0.7	1.44E+04	1.21E+05	0.036	2.21E+05	2.61E+06	0.634	6.42E+06	1.80E+08	16.392
0.8	1.97E+04	1.73E+05	0.053	3.39E+05	3.80E+06	1.008	7.75E+06	1.67E+08	22.698
0.9	3.14E+04	2.79E+05	0.085	8.71E+05	1.14E+07	2.458	1.57E+07	1.92E+08	61.131

Table 6
Average (Avg.) and maximum (Max.) error ratio.

	$n=10$, buffer=16,000 KB						$n=10$, buffer=30,720 KB					
	$\alpha C_{max}+(1-\alpha)T_{max}$			$\alpha C_{max}+(1-\alpha)\Sigma T_i$			$\alpha C_{max}+(1-\alpha)T_{max}$			$\alpha C_{max}+(1-\alpha)\Sigma T_i$		
α	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT
0.1	1.697	7.333	12	1.819	8.175	12	1.127	4.940	21	1.146	6.995	22
0.2	1.631	7.333	15	1.710	7.333	16	1.089	6.543	23	1.066	6.000	25
0.3	1.923	8.833	16	2.065	8.833	17	1.185	7.536	21	1.199	6.000	20
0.4	2.136	8.872	18	2.195	9.474	17	1.277	8.154	21	1.233	6.000	21
0.5	2.122	8.500	18	2.140	8.500	17	1.240	5.419	24	1.530	8.576	23
0.6	2.425	15.722	14	2.561	15.722	14	1.424	8.436	21	1.441	8.883	21
0.7	2.495	13.571	15	2.559	15.048	15	1.352	6.486	20	1.629	9.115	21
0.8	2.344	8.958	14	2.407	10.708	13	1.519	10.508	16	2.047	8.008	15
0.9	1.844	5.745	13	1.950	6.148	11	1.158	4.593	15	1.965	8.284	14
	$n=12$, buffer=16,000 KB						$n=12$, buffer=30,720 KB					
	$\alpha C_{max}+(1-\alpha)T_{max}$			$\alpha C_{max}+(1-\alpha)\Sigma T_i$			$\alpha C_{max}+(1-\alpha)T_{max}$			$\alpha C_{max}+(1-\alpha)\Sigma T_i$		
α	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT
0.1	1.914	11.936	11	1.770	11.936	10	0.968	7.912	22	1.256	8.793	17
0.2	1.974	10.345	11	2.027	10.758	12	0.992	7.407	23	1.203	7.510	21
0.3	2.257	14.147	12	2.181	14.147	11	1.039	7.298	24	1.235	7.298	19
0.4	2.143	10.942	11	2.283	10.942	12	1.093	6.856	23	1.280	5.823	18
0.5	2.495	19.401	12	2.395	10.128	11	1.137	7.005	22	1.362	5.901	18
0.6	2.417	14.123	10	2.561	15.739	10	1.235	9.035	18	1.460	7.912	14
0.7	2.496	13.634	10	2.871	11.392	9	1.304	6.988	19	1.710	6.988	15
0.8	2.515	11.174	10	2.733	12.019	7	1.499	6.760	14	1.814	8.022	11
0.9	2.195	7.486	10	2.336	9.254	6	1.168	6.988	7	1.634	6.988	7
	$n=14$, buffer=16,000 KB						$n=14$, buffer=30,720 KB					
	$\alpha C_{max}+(1-\alpha)T_{max}$			$\alpha C_{max}+(1-\alpha)\Sigma T_i$			$\alpha C_{max}+(1-\alpha)T_{max}$			$\alpha C_{max}+(1-\alpha)\Sigma T_i$		
α	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT	Avg.	Max.	UB=OPT
0.1	2.151	7.443	3	2.513	9.238	5	1.473	8.477	18	1.615	8.477	17
0.2	2.349	7.861	5	2.790	9.663	7	1.148	10.596	18	1.708	11.71	20
0.3	2.633	8.645	5	3.150	10.581	7	1.236	9.272	18	1.778	9.272	15
0.4	2.729	8.445	5	3.116	10.061	6	1.315	8.344	17	2.129	10.462	13
0.5	2.730	8.485	3	3.245	12.539	3	1.437	8.212	17	2.223	9.504	13
0.6	2.882	8.035	2	3.394	12.990	2	1.486	10.331	15	2.076	10.331	11
0.7	2.671	6.288	3	3.179	10.312	4	1.492	9.499	14	1.872	9.745	10
0.8	2.625	6.917	1	3.125	8.380	1	1.396	8.015	15	1.708	8.015	10
0.9	2.084	6.547	2	2.786	12.078	2	1.187	7.682	12	1.627	9.978	7

objects based on a user's query. Each object has certain metadata describing its contents, file size, playback duration, and due date constraints.

2) *Bandwidth Estimation Module.* The main function of this module is to estimate the average available end-to-end bandwidth in a presentation session. The bandwidth is used to estimate the transmission durations of the requested media objects. In the past, researchers have worked to obtain accurate end-to-end measurement algorithms for available bandwidth. A variety of techniques (refer to [28] for a survey of different approaches) can be applied to estimate the average bandwidth for which the transmission duration can be approximated based on the file size of an object. In principle, an accurate estimation of bandwidth is crucial to providing the optimal schedule of the media objects. Nevertheless, an optimal schedule computation based on a rough estimation of the bandwidth could still significantly reduce the overall presentation lag as compared to random sequences. In the evaluation experiments, we will provide results showing such a situation.
3) *Scheduling Engine.* The Scheduling Engine computes the optimal sequence based on the estimated bandwidth, the given constraints, the file sizes, and playback durations of the selected media objects. Depending on the required response times of the online applications, the final sequence could be the optimal solution for problems with a small number of objects, or a near-optimal solution when the server needs to serve a large number of clients in real time. The final sequence is a "playlist" that will be sent to the client.
4) *Object Prefetcher.* Based on the playlist, Object Prefetcher requests and transmits the media objects under a typical HTTP protocol. For applications with a client-side buffer constraint, the presentation status

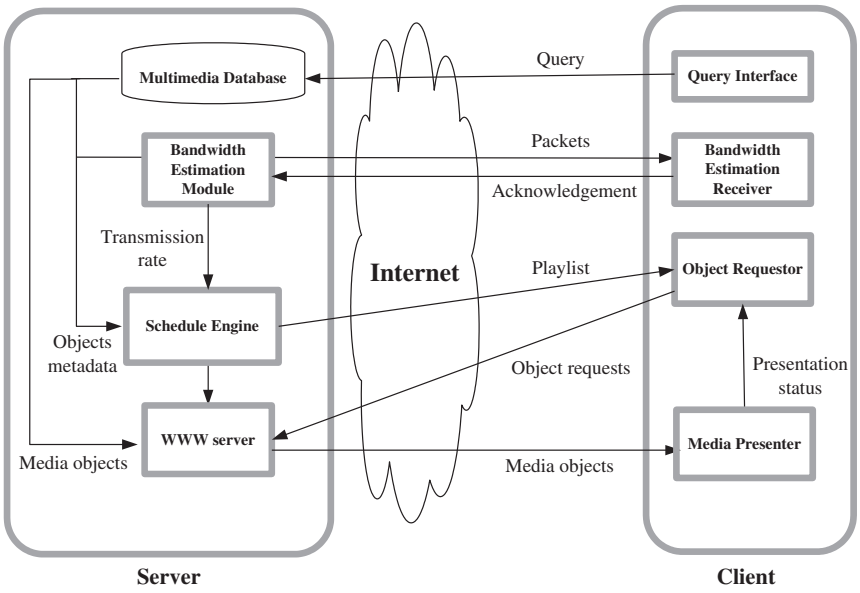

Fig. 8. System architecture of the prototype media delivery and presentation system.

and buffer status detected from Media Presenter are used to decide the exact time to prefetch an object.

5) *Media Presenter*. The major function of Media Presenter is to integrate and continuously play the retrieved media objects. Furthermore, during a presentation, it reports the playback and buffer status to Object Prefetcher.

We provided a prototype system implementation using Macromedia Flash technology. Fig. 9 shows a snapshot of a Flash-based prefetch-enabled media player supporting TV-like presentations for desktop platforms. Fig. 10 shows two snapshots of the Flash-based media players for the PDA platforms. The bandwidth estimation algorithm applied was based on a naive approach where the server sends a small file to the client, and the client sends an acknowledgment signal once the file is completely downloaded. The bandwidth is given as the ratio of file size to the corresponding transmission time.

Using the prototype system implementation, this section presents experiments for evaluating the proposed MNEH heuristic algorithm in real-world network environments with certain network fluctuations. We compared the objective function values (i.e., the weighted sum of the presentation lag and due date penalties) of job sequences yielded by the proposed MNEH heuristic algorithm and the *EDD* dispatching rule. The experiments investigated cases with 20% due date-constrained objects in the presentation. The media objects in the server were delivered to a PDA client through a public 3 G wireless network using WCDMA technology [13] with a maximum bandwidth of 384 k bytes/s.

In the first experiment, we randomly selected 10 sets of media objects from an educational media archive on our campus. Each data set contains 10 different media objects with different file sizes and playback durations. In the second experiment, each dataset contains 20 objects. The buffer size of the media players in the PDA was set to 8 Mbytes.

For each data set, we conducted 3 runs with random sequences and another 3 runs with sequences computed by the proposed MNEH heuristic algorithm based on the bandwidth estimated immediately before each presentation session. For each run, the transmission time and playback time of the objects through the WCDMA network were recorded. The values of the objective function $\alpha C_{max}+(1-\alpha)T_{max}$ with $\alpha=0.5$ were calculated accordingly.

Figs. 11 and 12 show the average objective function values of the *EDD*-based sequences and the proposed MNEH sequences for the first experiment (10 objects) and the second experiment (20 objects), respectively. The results for both experiments signify a clear reduction of up to 20% in the average objective function values if the proposed MNEH heuristic algorithm was applied.

8. Conclusions and future work

In this study, we formulated the problem of sequencing media objects in a dynamically composed multimedia presentation from digital libraries so as to reduce the overall presentation lag and due date penalties. The addressed problem was mapped to two-machine flowshop scheduling with buffer and due date constraints. Two different bi-criteria objective functions, namely $\alpha C_{max}+(1-\alpha)T_{max}$ and $\alpha C_{max}+(1-\alpha)\Sigma T_i$, are considered. Exact and approximation algorithms were proposed and implemented in our system for providing digital library services. Experimental results show that the proposed methods can obtain optimal solutions for problems with up to 14 objects with a small buffer, and 16 objects with a large buffer. Furthermore, the proposed NEH-based heuristic solutions appeared to provide quality approximate solutions with an average error rate of under 4%. The heuristic algorithm can be applied to heavily-loaded servers that require a quick response on the

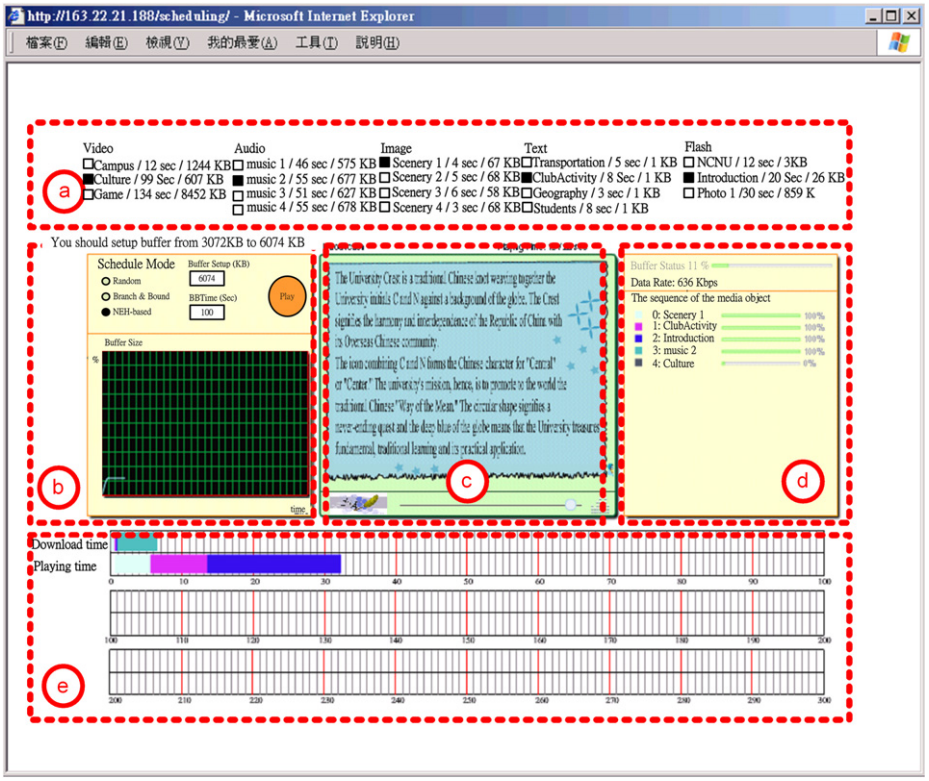

Fig. 9. Snapshot of a Flash-based prefetch-enabled media player for Desktop platforms, (a) retrieved objects, (b) buffer status, (c) the presentation of the object currently under playback, (d) transmission status of each object, (e) Gantt chart.

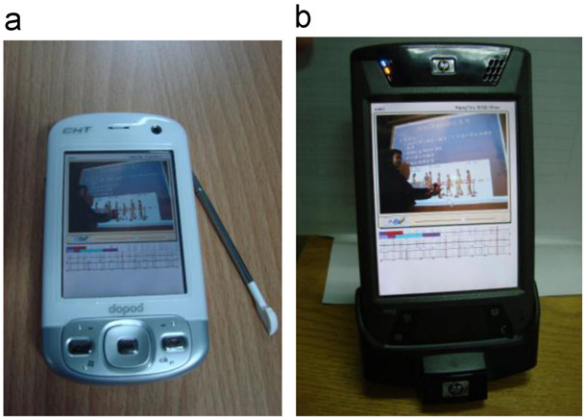

Fig. 10. Snapshot of a Flash-based prefetch-enabled media player supporting TV-like presentations for (a) 3 G PDA phone, (b) PDA platform.

near-optimal sequence of a media set with more than a dozen objects. Experiments with WCDMA wireless networks indicate that the proposed MNEH heuristic algorithm can produce solutions whose objective function values are in general smaller than the solutions yielded by the *EDD* dispatching rule.

Following this study, there are some potential research directions. The studied settings assumed that the end-to-end transmission rate is deterministic. This assumption can be regarded an acceptable approximation for applications in which the end-to-end transmission delay is mainly subject to the last mile bottleneck without significant bandwidth variations. However, many Internet applications provide only best-effort data delivery and cannot guarantee a very consistent transmission rate during a presentation session. In such cases, stochastic variations of the network transmission rate (or the download time of a media object) should be considered in the sequence optimization process. In the literature, stochastic flowshop scheduling problems deal with cases for which the processing times of jobs on each machine are governed by a random variable. In general, the stochastic flowshop scheduling problems are not all that easy to solve. Most of the existing studies focus on problems where the job processing time is a random variable following an "exponential distribution". The objective is to find the schedule that minimizes the expected makespan $E(C_{max})$. Such a case with exponentially distributed process times has been shown to be, in general, tractable ([34]; Bagga [3]; Cunningham and Dutta [7]; and Ku and Niu [19]). Unfortunately, for Internet multimedia delivery applications, the probability density function of an end-to-end bandwidth is, typically, inverse bell-shaped [14]. The probability density function of the download time of a media object should resemble a truncated normal distribution or an Erlang distribution with high shape parameters. Two-machine flowshop scheduling problems with these two distributions have never been solved and

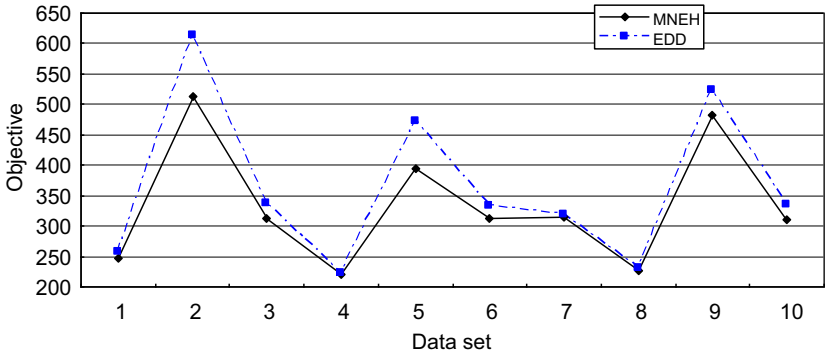

Fig. 11. Average objective values of *EDD*-based sequences and MNEH-based sequences. Each media set contains 10 objects.

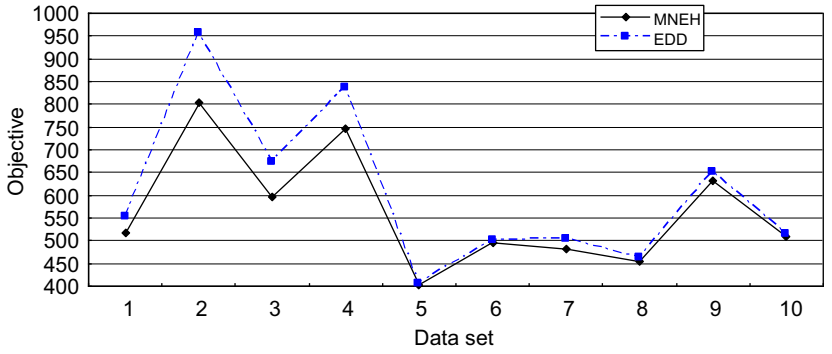

Fig. 12. Average objective values of *EDD*-based sequences and MNEH-based sequences. Each media set contains 20 objects.

thus are worthy of research attention. In addition, in many applications the media objects in a digital library are subject to different categories of temporal constraints (e.g., sequence, parallel, precedence). Development of exact or approximate algorithms for these problems requires further investigation.

Acknowledgments

Feng-Cheng Lin was supported by the National Science Council of Taiwan, under grant NSC 100-2625-M-035-003 and NSC 100-2119-M-035-001. Jen-Shin Hong was supported by the National Science Council of Taiwan, under grant NSC 96-2221-E-260-020-MY3. Bertrand M.T. Lin was supported in part by the National Science Council of Taiwan, under grant NSC 96-2416-H-009-012-MY2.

References

[1] A. Agarwal, S. Colak, E. Eryarsoy, Improvement heuristic for the flow-shop scheduling problem: an adaptive-learning approach, European Journal of Operational Research 169 (2006) 801–815.
[2] A. Allahverdi, A new heuristic for *m*-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness, Computers & Operations Research 31 (2004) 157–180.
[3] P.C. Bagga, *N*-job, 2-machine sequencing problem with stochastic service times, OpSearch 7 (1970) 184–197.
[4] P. Brucker, Scheduling Algorithms, third ed. Springer-Verlag, Berlin, 2001.
[5] P. Brucker, S. Heitmann, J. Hurink, Flow-shop problems with intermediate buffers, OR Spectrum 25 (2003) 549–574.
[6] K. Chakravarthy, C. Rajendran, A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization, Production Planning & Control 10 (1999) 707–714.
[7] A.A. Cunningham, S.K. Dutta, Scheduling jobs with exponentially distributed processing times on a flow shop, Naval Research Logistics Quarterly 16 (1) (1973) 69–81.
[8] R.L. Daniel, R.J. Chambers, Multi-objective flow shop scheduling, Naval Research Logistics Quarterly 37 (1990) 981–995.
[9] S.K. Dutta, A.A. Cunningham, Sequencing two-machine flow-shops with finite intermediate storage, Management Science 21 (1975) 989–996.
[10] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Franciso, CA, 1979.
[11] P.C. Gilmore, R.E. Gomory, Sequencing a one-state variable machine: a solvable case of the traveling salesman problem, Operations Research 12 (1964) 665–679.
[12] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics 5 (1979) 287–326.
[13] H. Honkasalo, K. Pehkonen, M.T. Niemi, A.T. Leino, WCDMA and WLAN for 3 G and beyond, IEEE Wireless Communications 9 (2) (2002) 14–18.
[14] S.C. Hui, J.Y.B. Lee, On available bandwidth in many-to-one data transfer—modeling and applications, Multimedia Tools and Applications 34 (2) (2007) 139–154.
[15] J.R. Jackson, Scheduling a Production Line to Minimize Maximum Tardiness, Management Science Research Project, Research Report 43, University of California, Los Angeles, 1955.
[16] S.M. Johnson, Optimal two- and three-stage production schedules with setup time included, Naval Research Logistics Quarterly 1 (1954) 61–68.
[17] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer-Verlag, Berlin, 2004.
[18] Y. Kim, A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops, Computers & Operations Research 20 (1993) 391–401.
[19] P.-S. Ku, S.-C. Niu, On Johnson's two-machine flow shop with random processing times, Operations Research 34 (1) (1986) 130–136.

[20] R. Leisten, Flowshop sequencing problems with limited buffer storage, International Journal of Production Research 28 (11) (1990) 2085–2100.
[21] J. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problem, Annals of Discrete Mathematics 1 (1977) 343–362.
[22] F.C. Lin, C.Y. Lai, J.S. Hong, Minimize presentation lag by sequencing media objects for auto-assembled presentations from digital libraries, Data and Knowledge Engineering 66 (3) (2008) 382–401.
[23] F.C. Lin, J.S. Hong, B.M.T. Lin, A two-machine flowshop problem with processing time-dependent buffer constraints—an application in multimedia presentation, Computers & Operations Research 36 (4) (2009) 1158–1175.
[24] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, Wiley, Chichester, 1990.
[25] M. Nawaz, E. Enscore, I.A. Ham, A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, Omega 11 (1983) 91–95.
[26] J.C.H. Pan, E.T. Fan, Two-machine flow-shop scheduling to minimize total tardiness, International Journal of Systems Science 29 (1997) 405–414.
[27] J.C.H. Pan, J.S. Chen, C.M. Chao, Minimizing tardiness in a two-machine flow-shop, Computers & Operations Research 29 (2002) 869–885.
[28] R.S. Prasad, M. Murray, C. Dovrolis, K. Claffy, Bandwidth estimation: metrics, measurement techniques, and tools, IEEE Network 17 (6) (2003) 27–35.
[29] C.H. Papadimitriou, P.C. Kanellakis, Flowshop scheduling with limited temporary storage, Journal of the Association for Computing Machinery 27 (1980) 533–549.
[30] R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flowshop heuristic, European Journal of Operational Research 165 (2005) 479–494.
[31] T. Sen, P. Dileepan, J.N.D. Gupta, The two-machine flowshop scheduling problem with total tardiness, Computers & Operations Research 16 (1989) 333–340.
[32] C. Smutnicki, A two-machine permutation flow shop scheduling problem with buffers, OR Spectrum 20 (1998) 229–235.
[33] L. Tang, H. Xuan, Lagrangian relaxation algorithms for real-time hybrid flowshop scheduling with finite intermediate buffers, Journal of the Operational Research Society 57 (2006) 316–324.
[34] T.T. Talwer, A note on sequencing problems with uncertain job time, Journal of the Operational Research Society of Japan 9 (1967) 93–97.
[35] L. Wang, L. Zhang, D.-Z. Zheng, An effective hybrid genetic algorithm for flow shop scheduling with limited buffers, Computers & Operations Research 33 (2006) 2960–2971.