Discrete Event Dyn Syst (2009) 19:267–282
DOI 10.1007/s10626-009-0061-z

A Subgradient Descent Algorithm for Optimization of Initially Controllable Flow Shop Systems

Kagan Gokbayrak · Omer Selvi

Received: 7 September 2007 / Accepted: 13 January 2009 /
Published online: 4 February 2009

Abstract We consider an optimization problem for deterministic flow shop systems processing identical jobs. The service times are initially controllable; they can only be set before processing the first job, and cannot be altered between processes. We derive some waiting and completion time characteristics for fixed service time flow shop systems, independent of the cost formulation. Exploiting these characteristics, an equivalent convex optimization problem, which is non-differentiable, is derived along with its subgradient descent solution algorithm. This algorithm not only eliminates the need for convex programming solvers but also allows for the solution of larger systems due to its smaller memory requirements. Significant improvements in solution times are also observed in the numerical examples.

Keywords Convex optimization · Subgradient descent algorithm · Initially controllable service times · Flow shop

1 Introduction

We consider deterministic serial manufacturing systems processing identical jobs with given arrival times. The queues of the machines are unlimited in size and operate under the first-in-first-out (FIFO) discipline. The machines are manually controllable as opposed to CNC (Computer Numerical Control) machines considered in Gokbayrak and Selvi (2007). During mass production, it may not be feasible to alter the service times of these machines because the setup times are idle times and the manual modifications are prone to errors. Therefore, we assume that the service

K. Gokbayrak (✉) · O. Selvi
Department of Industrial Engineering, Bilkent University, Ankara, 06800, Turkey
e-mail: kgokbayr@bilkent.edu.tr

O. Selvi
e-mail: selvi@bilkent.edu.tr

times at machines are initially controllable; they are set before the arrival of the first job and cannot be altered afterwards.

The objective of this study is to minimize a cost function composed of service costs on machines, which are dependent on service times, and regular completion time costs for jobs, which for a special case account for inventory holding costs. Motivated by the extended Taylor's tool-wear equation in Kalpakjian and Schmid (2006), we assume that a smaller service time incurs a higher cost because a faster service increases wear and tear on the tools due to increased temperatures and may need extra supervision. A slower service, on the other hand, builds up inventory and postpones the completion times increasing the completion time costs. This trade-off in setting the service times makes the problem nontrivial. The job sequencing problem, which is known to be NP-hard even for fixed service times (see in Pinedo (2002)), is not considered and the objective of this study is limited to determining the cost minimizing service times to be used in the flow shop.

The idea of treating scheduling problems for deterministic machines as optimal control problems of discrete event dynamic systems first appeared in Gazarik and Wardi (1998) where job release times to a single machine were controlled to minimize the discrepancy between completion times and due dates. Following this work, service time control problems, where the service times can be adjusted between processes, were considered: Pepyne and Cassandras (1998) formulated an optimal control problem for a single machine system with the objective of completing jobs as fast as possible with the least amount of control effort. In Pepyne and Cassandras (2000), they extended their results to jobs with completion due dates penalizing both earliness and tardiness. The uniqueness of the optimal solution for the single stage control problem was shown in Cassandras et al. (2001). Exploiting the structural properties of the optimal sample path for the single machine problem, Wardi et al. (2001) and Cho et al. (2001) developed backward-in-time and forward-in-time solution algorithms, respectively. The forward-in-time algorithm was later improved by Zhang and Cassandras (2002). In a related work, Moon and Wardi (2005) considered a single machine problem where the completed jobs wait in a finite size output buffer until their due dates. They presented an efficient solution algorithm for this system with blocking.

Two machine problems with identical jobs were solved in Cassandras et al. (1999) using a Bezier approximation method. Gokbayrak and Cassandras (2000) and Gokbayrak and Selvi (2006) identified optimal sample path characteristics for two machine problems. Finally, Gokbayrak and Selvi (2007) considered multimachine flow shop systems with regular costs on completion times and decreasing costs on service times. The resulting optimization problem was non-convex and non-differentiable. It was shown that, on the optimal sample path, jobs do not wait between machines, a property which allowed for simple convex programming formulations. Under strict convexity assumptions, a forward-in-time solution algorithm was developed.

On a different line of work, we replaced the CNC machines in the flow shop by traditional non-CNC machines: Gokbayrak and Selvi (2008) considered the system in Gokbayrak and Selvi (2007) with the additional constraint that the service times are initially controllable. Even though this seemed to be a simple modification, since the no-waiting property does not hold in the flow shop systems with fixed service times, the analysis was changed completely. We derived some waiting and completion

time characteristics in these systems, independent of the optimization problem, and exploited them to derive a simpler equivalent convex optimization problem. Even though the resulting problem formulation enables solutions for large systems, it still needs the use of a solver which may not be available at some of the manufacturing companies. The need for a lower cost optimization tool motivated us for the work presented in this paper. We continue along the same lines to obtain additional waiting and completion time characteristics, and derive another equivalent convex optimization problem, which is non-differentiable. A subgradient descent algorithm is also developed for solving this optimization problem. This algorithm eliminates the need for a solver and has considerably low memory requirements, therefore, it allows to solve optimization problems of even larger systems.

The rest of the paper is organized as follows: In Section 2, we formulate a non-convex and non-differentiable optimization problem. In this section, we also present the equivalent convex optimization formulation obtained in Gokbayrak and Selvi (2008). Section 3 presents some waiting and completion time characteristics in fixed service time flow shop systems, independent of the objective function. Exploiting these characteristics, an equivalent convex optimization problem is derived in Section 4 along with a subgradient descent algorithm with projections. Section 5 presents numerical examples to illustrate the performance of the solution algorithm and to verify the waiting and completion time characteristics derived earlier. Finally, Section 6 concludes the paper.

2 Problem formulation

We consider a sequence of N identical jobs, denoted by $\{C_i\}_{i=1}^{N}$, arriving at an M-machine flow shop system at known times $0 \leq a_1 \leq a_2 \leq ... \leq a_N$. Machines process one job at a time on a FIFO non-preemptive basis (i.e. a job in service can not be interrupted until its service completion). The buffers in front of the machines are assumed to be of infinite sizes.

We define a temporal state $x_{i,j}$ that keeps the departure time information of job C_i from machine j. The relationships between the temporal states are given by the following max-plus equations (see in Cassandras and Lafortune (1999))

$$x_{i,j} = \max(x_{i,j-1}, x_{i-1,j}) + s_j \tag{1}$$

$$x_{i,0} = a_i \,, \quad x_{0,j} = -\infty \tag{2}$$

for $i = 1, ..., N$ and $j = 1, ..., M$, where the service time at machine $j \in \{1, ..., M\}$ is denoted by s_j. Note that, unlike the system considered in Gokbayrak and Selvi (2007) where the service times could differ from one job to another, the same service time s_j is applied to all jobs at machine j.

The discrete-event optimal control problem, denoted by P, is the determination of the optimal service times

$$P: \min_{\substack{s_j \geq S_j \\ j=1,...,M}} \left\{ J = \sum_{j=1}^{M} \theta_j(s_j) + \sum_{i=1}^{N} \phi_i(x_{i,M}) \right\} \tag{3}$$

subject to Eqs. 1 and 2 for $i = 1, ..., N$ and $j = 1, ..., M$. In this formulation, θ_j denotes the total process cost over all jobs at machine j, and ϕ_i denotes the completion time cost for job C_i. The minimum service time required at machine j, a physical constraint dictated by the machine or the process dynamics, is denoted by S_j.

The following assumptions are necessary to make the problem somewhat more tractable while preserving the originality of the problem.

Assumption 1 $\theta_j(\cdot)$, for $j = 1, ..., M$, is continuously differentiable, monotonically decreasing and convex.

Assumption 2 $\phi_i(\cdot)$, for $i = 1, ..., N$, is continuously differentiable, monotonically increasing and convex.

These assumptions indicate that longer services will decrease the service costs while increasing the departure times, hence the completion time costs.

Due to the *max* function in Eq. 1, P is non-convex. Exploiting some temporal state characteristics and linearizing the *max* functions in the constraints, the following equivalent convex optimization problem was derived in Gokbayrak and Selvi (2008):

$$Q: \min_{\substack{s_j, x_{i,M} \\ i=1,...,N \\ j=1,...,M}} \left\{ J = \sum_{j=1}^{M} \theta_j(s_j) + \sum_{i=1}^{N} \phi_i(x_{i,M}) \right\} \tag{4}$$

subject to

$$x_{1,M} = a_1 + \sum_{k=1}^{M} s_k \tag{5}$$

$$x_{i,M} \geq a_i + \sum_{k=1}^{M} s_k \tag{6}$$

$$x_{i,M} \geq x_{i-1,M} + s_j \tag{7}$$

$$s_j \geq S_j \tag{8}$$

for all $i = 2, ..., N$ and $j = 1, ..., M$. This formulation enabled us to solve optimization problems of large systems utilizing commercial convex programming problem solvers. The motivation for the study in this paper, however, is to be able to solve such problems without the commercial solvers.

In the next section, we present temporal state characteristics of the flow shop systems with fixed service times. Exploiting these characteristics, in Section 4, we derive another equivalent convex optimization problem with fewer decision variables and no constraints (except for the physical constraints on the service times.)

3 Temporal state characteristics of the system

In our flow shop system, each machine j performs some service of duration s_j. Based on these service times, we define the following:

Definition 1 Machine u is a *local bottleneck* if its service time exceeds the service times of all upstream machines, i.e., $s_u > \max_{j=0,\ldots,u-1} s_j$ where s_0 is defined to be zero.

Since the first machine is a local bottleneck, there is at least one local bottleneck in every flow shop system.

Definition 2 Machines $\{u, \ldots, v\}$ form a *flushing portion* if

1) Machine u is a local bottleneck, i.e., $s_u > \max_{j=0,\ldots,u-1} s_j$
2) There are no local bottlenecks in machines $\{u+1, \ldots, v\}$, i.e., $s_u \geq \max_{j=u+1,\ldots,v} s_j$
3) If $v < M$, then machine $(v+1)$ is a local bottleneck, i.e., $s_u < s_{v+1}$.

Every local bottleneck starts a flushing portion, and the last flushing portion is ended by machine M.

We borrow the following two lemmas from Gokbayrak and Selvi (2008) to establish the waiting characteristics at machines.

The first lemma establishes that jobs may wait at only the local bottlenecks.

Lemma 1 *In a flushing portion, no-waiting is observed after its local bottleneck machine.*

The second lemma suggests that, given the waiting status of a job at a local bottleneck machine, we may deduce its waiting status at a downstream or an upstream local bottleneck machine.

Lemma 2 *If job C_i waits for service at some local bottleneck, then it will wait for service at all downstream local bottlenecks.*

As it turns out, waiting is observed only at the local bottleneck machines. Given the arrival times of the jobs and the service time of some local bottleneck machine u, we can determine which jobs wait at this machine. Let us define the average interarrival time between jobs C_k and C_l , where $k > l$ as

$$\sigma_k^l = \frac{a_k - a_l}{k - l} \tag{9}$$

The minimum of the average interarrival times for job C_k is, then, defined as

$$\sigma_k = \begin{cases} \infty & k = 1 \\ \min\limits_{l=1,\ldots,k-1} \sigma_k^l & k > 1 \end{cases} \tag{10}$$

The following lemma allows us to determine whether a job waits or not at some local bottleneck machine u.

Lemma 3 *A job C_k waits for service at the local bottleneck machine u if and only if $\sigma_k < s_u$.*

Proof (Necessity) Let us assume that C_k does not wait at the local bottleneck machine u. According to Lemmas 1 and 2, no waiting is observed by the job at the upstream machines, therefore we have

$$x_{k,u} = a_k + \sum_{j=1}^{u} s_j \tag{11}$$

For previous jobs $\{C_i\}_{i=1}^{k-1}$, we can write

$$x_{i,u} \geq a_i + \sum_{j=1}^{u} s_j \tag{12}$$

Hence, from Eqs. 11 and 12, we get

$$x_{k,u} - x_{i,u} \leq a_k - a_i \tag{13}$$

for $i = 1, ..., k-1$. Since the departure times (from machine u) of two consecutive jobs are at least s_u apart, we can write

$$x_{k,u} - x_{i,u} \geq (k-i)s_u \tag{14}$$

for $i = 1, ..., k-1$. From Eqs. 9, 13, and 14, we have

$$\sigma_k^i \geq s_u$$

for all $i = 1, ..., k-1$, resulting in, from Eq. 10,

$$\sigma_k \geq s_u$$

(Sufficiency) Let us assume that job C_k waits at machine u. Then, we have

$$x_{k,u} > a_k + \sum_{j=1}^{u} s_j \tag{15}$$

Let C_i be the last job in $\{C_1, ..., C_{k-1}\}$ that does not wait at machine u (since job C_1 does not wait at any machine, existence of such a job is guaranteed.) Then, according to Lemmas 1 and 2, C_i does not wait at any upstream machines, so we can write

$$\begin{aligned} x_{k,u} &= x_{i,u} + (k-i)s_u \\ &= a_i + \sum_{j=1}^{u} s_j + (k-i)s_u \end{aligned} \tag{16}$$

From Eqs. 9, 15, and 16, we get

$$\sigma_k^i < s_u$$

resulting in, from Eq. 10,

$$\sigma_k < s_u$$

□

We describe the waiting characteristics of jobs at local bottleneck machines by block structures.

Definition 3 A contiguous set of jobs $\{C_i\}_{i=k}^{n}$ is said to form a *block* at a local bottleneck machine u if

1) Jobs C_k and C_{n+1} (if exists) do not wait at machine u, i.e., $x_{k-1,u} \leq x_{k,u-1}$ and $x_{n,u} \leq x_{n+1,u-1}$ for $n < N$.
2) Jobs $\{C_i\}_{i=k+1}^{n}$ wait at machine u, i.e., $x_{i-1,u} > x_{i,u-1}$ for $i = k+1, ..., n$.

Each block starts with a non-waiting job k and continues with waiting jobs $\{C_i\}_{i=k+1}^{n}$with departure times

$$x_{i,u} = x_{k,u} + (i-k)s_u \tag{17}$$

Definition 4 A partition of jobs into blocks is called a *block structure*.

For any given service time s_u, by modifying the arrival times, we can generate 2^N different block structures at a local bottleneck machine u . If the arrival times are given, however, by modifying the service time s_u, we can generate at most N different block structures. The next lemma establishes this upper bound on the number of different block structures at a local bottleneck machine.

Lemma 4 *There are at most N different block structures at any local bottleneck machine u.*

Proof From Lemma 3, a job C_i starts a block at a local bottleneck machine u iff $\sigma_i \geq s_u$. Reindexing σ_i's as

$$\sigma_{(1)} \leq \sigma_{(2)} \leq ... \leq \sigma_{(N)}$$

each interval $(\sigma_{(k-1)}, \sigma_{(k)}]$, where $\sigma_{(0)} = 0$, defines a block structure: If $s_u \in (\sigma_{(k-1)}, \sigma_{(k)}]$, then all jobs in the set $\{C_i : \sigma_i \geq \sigma_{(k)}\}$ start blocks at machine u while others do not. Since there are at most N such intervals, there are at most N different block structures. □

According to Lemma 3, one could evaluate σ_k values for all jobs C_k and compare them to the service time of the local bottleneck machine to determine the block structure. The following lemma, however, presents a computationally simpler way to determine the block structure, which is implemented in the subgradient descent algorithm in the next section.

Lemma 5 *If jobs $\{C_i\}_{i=k}^{n}$ form a block at machine u, then,*

$$\sigma_i^k < s_u$$

is satisfied for all $i = k+1, ..., n$.

Proof (By Induction) Since C_k starts the block, we know by definition that it does not wait at machine u. Hence, from Lemma 3, we have $\sigma_k \geq s_u$, i.e., for all $l < k$, we can write

$$\sigma_k^l = \frac{a_k - a_l}{k-l} \geq s_u \tag{18}$$

In order to show the basis step by a contradiction, we assume that

$$\sigma_{k+1}^{k} = a_{k+1} - a_k \geq s_u \tag{19}$$

From Eqs. 18 and 19, we get for all $l < k$

$$\begin{aligned}\sigma_{k+1}^{l} &= \frac{a_{k+1} - a_l}{k+1-l} \\ &= \frac{(a_{k+1} - a_k) + (a_k - a_l)}{k+1-l} \\ &\geq \frac{s_u + (k-l)s_u}{k+1-l} = s_u\end{aligned}$$

resulting in $\sigma_{k+1} \geq s_u$, which contradicts, from Lemma 3, that job C_{k+1}waits.

In order to show the induction step again by contradiction, we assume that

$$\sigma_i^k < s_u \tag{20}$$

for $i = k+1, ..., t-1$, where $t \leq n$ and

$$\sigma_t^k \geq s_u \tag{21}$$

From Eqs. 18 and 21, we have

$$\begin{aligned}\sigma_t^l &= \frac{a_t - a_l}{t-l} = \frac{(a_t - a_k) + (a_k - a_l)}{t-l} \\ &\geq \frac{(t-k)s_u + (k-l)s_u}{t-l} = s_u\end{aligned}$$

for all $l = 1, ..., k-1$. Moreover, from Eqs. 20 and 21, we have

$$\begin{aligned}\sigma_t^i &= \frac{a_t - a_i}{t-i} = \frac{(a_t - a_k) - (a_i - a_k)}{t-i} \\ &\geq \frac{(t-k)s_u - (i-k)s_u}{t-i} = s_u\end{aligned}$$

for all $i = k+1, ..., t-1$. Hence, from Eq. 10, $\sigma_t \geq s_u$, which contradicts, from Lemma 3, that job C_t waits. □

Starting with the first job C_1, which starts the first block, this lemma can be iteratively applied to determine the block structure at any local bottleneck. For this task, all we need are the arrival times of the jobs and the service time of the local bottleneck.

Next, we define the most downstream local bottleneck of the system as the global bottleneck, and derive the completion times of jobs based on the block structure at the global bottleneck machine.

Definition 5 The local bottleneck machine u with the highest service time $s_u = \max_{j=1,...,M} s_j$ is the *global bottleneck*.

There can be no local bottleneck machines downstream to a global bottleneck, i.e., no waiting is observed after the global bottleneck machine. Hence, the completion times can be determined as presented in the next lemma.

Lemma 6 *Let jobs $\{C_i\}_{i=k}^n$ form a block at the global bottleneck machine m. Then, the completion times of these jobs are given as*

$$x_{i,M} = a_k + (i-k)s_m + \sum_{j=1}^{M} s_j \tag{22}$$

for $i = k, ..., n$.

Proof Machines $\{m, ..., M\}$ form the last flushing portion of the system. By Lemma 1, jobs do not wait after the global bottleneck machine m, hence the completion times of the jobs $\{C_i\}_{i=k}^n$ can be written as

$$x_{i,M} = x_{i,m} + \sum_{j=m+1}^{M} s_j \tag{23}$$

for $i = k, ..., n$. By Lemma 2, since C_k does not wait at the global bottleneck machine m, it observes no waiting at the upstream machines. Hence,

$$x_{k,m} = a_k + \sum_{j=1}^{m} s_j \tag{24}$$

For jobs $\{C_i\}_{i=k+1}^n$ that wait at the global bottleneck machine m, we have

$$x_{i,m} = x_{k,m} + (i-k)s_m \tag{25}$$

Hence, from Eqs. 23, 24, and 25, the completion times of the jobs $\{C_i\}_{i=k}^n$ are given as

$$x_{i,M} = a_k + (i-k)s_m + \sum_{j=1}^{M} s_j$$

□

Next, we exploit the characteristics obtained in this section to derive a *minmax* problem and present a subgradient descent algorithm with projections as its solution methodology.

4 Optimization of service times

Let us employ Lemma 6 to rewrite the optimization problem P as

$$\hat{P}: \min_{\substack{s_j \geq S_j \\ j=1,...,M}} \left\{ J(s) = \sum_{j=1}^{M} \theta_j(s_j) + \sum_{b=1}^{B(s)} \sum_{i=k^b(s)}^{n^b(s)} \phi_i \left(a_{k^b(s)} + s_{\text{total}} + \left(i - k^b(s)\right) s_{\max} \right) \right\} \tag{26}$$

where, given the service times s, $s_{\max} = \max_{j=1,...,M} s_j$ is the service time of the global bottleneck machine, $s_{\text{total}} = \sum_{j=1}^{M} s_j$ is the total service time, $B(s)$ is the number of blocks at the global bottleneck machine, $k^b(s)$ and $n^b(s)$ are the indices of the first and the last jobs of the bth block, respectively.

Let

$$J_l(s) = \sum_{j=1}^{M} \theta_j(s_j) + \sum_{b=1}^{B_l} \sum_{i=k_l^b}^{n_l^b} \phi_i \left(a_{k_l^b} + s_{\text{total}} + \left(i - k_l^b\right) s_{\max} \right) \tag{27}$$

be a cost function, where B_l is the number of blocks, k_l^b and n_l^b are the indices of the first and the last jobs, respectively, of the bth block at some global bottleneck whose service time falls in the interval $(\sigma_{(l-1)}, \sigma_{(l)}]$. Note that, by Assumptions 1 and 2, J_l is continuous and convex in the service times. From Lemma 4, there are at most N different block structures at the global bottleneck, hence we have at most N different cost functions of this form.

If $s_{\max}$ falls in the interval $(\sigma_{(l-1)}, \sigma_{(l)}]$, then we have $J(s) = J_l(s)$. In other words, the formulation of $J(s)$ differs from interval to interval. The next lemma shows that $J(s)$ can be written as the maximum of all these functions, yielding a *minmax* optimization problem.

Lemma 7 *The cost function $J_l(s)$ exceeds all other cost functions, i.e., $J_l(s) = \max_{t\in\{1,\ldots,N\}} J_t(s)$, when $s_{\max} \in (\sigma_{(l-1)}, \sigma_{(l)}]$.*

Proof Let us take an arbitrary job C_i, where $i \in \{1, \ldots, N\}$, and let job C_{k_l} start the block at the global bottleneck machine that job C_i resides in when the global bottleneck machine's service time falls in the interval $(\sigma_{(l-1)}, \sigma_{(l)}]$, i.e., when $s_{\max} \in (\sigma_{(l-1)}, \sigma_{(l)}]$. The completion time in this case is given by Lemma 6 as

$$x_{i,M}^l = a_{k_l} + s_{\text{total}} + (i - k_l) s_{\max} \tag{28}$$

Let us also take an arbitrary block structure corresponding to some interval $(\sigma_{(t-1)}, \sigma_{(t)}]$, and let C_{k_t} start the block at the global bottleneck machine that job C_i resides in. Similarly from Lemma 6, the completion time for this block structure is given as

$$x_{i,M}^t = a_{k_t} + s_{\text{total}} + (i - k_t) s_{\max} \tag{29}$$

Now, assume that $s_{\max} \in (\sigma_{(l-1)}, \sigma_{(l)}]$. We would like to compare $J_l(s)$ and $J_t(s)$ under this assumption.

From Eqs. 28 and 29, the completion times satisfy

$$x_{i,M}^l - x_{i,M}^t = \left(a_{k_l} - a_{k_t}\right) + (k_t - k_l) s_{\max} \tag{30}$$

There are three cases to consider:

Case 1: For $t = l$, from Eq. 30, we have $x_{i,M}^l = x_{i,M}^t$.

Case 2: For $t < l$, i.e., for $\sigma_{(t)} < \sigma_{(l)}$, by Lemma 3, $k_t \geq k_l$ because decreasing the service time of the global bottleneck has the effect of separating blocks into smaller blocks. If $k_t = k_l$, then from Eq. 30, $x_{i,M}^l = x_{i,M}^t$. If, on the other

hand, $k_t > k_l$, then job C_{k_t} is in the block started by job C_{k_l}, which leads to $\sigma_{k_t}^{k_l} < s_{\max}$ by Lemma 5. Therefore, we have, from Eqs. 9 and 30, that

$$\begin{aligned} x_{i,M}^{l} - x_{i,M}^{t} &= (k_t - k_l)\left[s_{\max} - \frac{(a_{k_t} - a_{k_l})}{(k_t - k_l)}\right] \\ &= (k_t - k_l)\left[s_{\max} - \sigma_{k_t}^{k_l}\right] \geq 0 \end{aligned}$$

Case 3: For $t > l$, i.e., for $\sigma_{(t)} > \sigma_{(l)}$, by Lemma 3, $k_t \leq k_l$ because increasing the service time of the global bottleneck has the effect of combining blocks into larger blocks. If $k_t = k_l$, then from Eq. 30, $x_{i,M}^{l} = x_{i,M}^{t}$. If, on the other hand, $k_t < k_l$, then since $\sigma_{k_l} > s_{\max}$ by Lemma 3, we have, from Eqs. 9, 10, and 30, that

$$\begin{aligned} x_{i,M}^{l} - x_{i,M}^{t} &= (k_t - k_l)\left[s_{\max} - \frac{(a_{k_l} - a_{k_t})}{(k_l - k_t)}\right] \\ &= (k_l - k_t)\left[\sigma_{k_l}^{k_t} - s_{\max}\right] \\ &\geq (k_l - k_t)\left[\sigma_{k_l} - s_{\max}\right] \geq 0 \end{aligned}$$

Hence, from all three cases, $x_{i,M}^{l} \geq x_{i,M}^{t}$, when $s_{\max} \in (\sigma_{(l-1)}, \sigma_{(l)}]$. By Assumption 2, ϕ_i is monotonically increasing, therefore, from Eq. 22 and 27,

$$J_l(s) - J_t(s) = \sum_{i=1}^{N} \phi_i\left(x_{i,M}^{l}\right) - \phi_i\left(x_{i,M}^{t}\right) \geq 0$$

Since $t \leq N$ is arbitrary, the result follows. □

Hence, from Lemma 7, we can write the optimization problem as

$$R: \min_{\substack{s_j \geq S_j \\ j=1,\ldots,M}} \left\{ J_R = \max_{l} J_l(s) \right\} \tag{31}$$

where J_l is the convex and continuous cost function corresponding to the interval $(\sigma_{(l-1)}, \sigma_{(l)}]$. Being the maximum of convex and continuous functions, J_R is a convex and continuous function of the service times.

4.1 Subgradient descent algorithm with projections

According to Lemma 7, when the global bottleneck machine's service time $s_{\max}$ falls in an interval $(\sigma_{(l-1)}, \sigma_{(l)}]$ for some $l \leq N$, the cost is $J_R = J_l(s)$. Therefore, for this case, the sensitivities of J_R to service times (at differentiable points) can be written as

$$\frac{\partial J_R}{\partial s_j} = \begin{cases} \theta_j'(s_j) + \sum_{b=1}^{B_l} \sum_{i=k_l^b}^{n_l^b} \phi_i'\left(a_{k_l^b} + s_{\text{total}} + \left(i - k_l^b\right) s_{\max}\right) & s_j < s_{\max} \\ \theta_j'(s_{\max}) + \sum_{b=1}^{B_l} \sum_{i=k_l^b}^{n_l^b} \left[\phi_i'\left(a_{k_l^b} + s_{\text{total}} + \left(i - k_l^b\right) s_{\max}\right)\left(1 + i - k_l^b\right)\right] & \sigma_{(l)} > s_j > \max_{i \neq j} s_i \end{cases} \tag{32}$$

for $j = 1, \ldots, M$. Note that when $s_j = \sigma_{(l)}$, i.e., when the block structure at the global bottleneck machine is about to change, or when $s_j = \max_{i \neq j} s_i$, i.e., when there are

other machines with the maximum service time, non-differentiability is observed. For these machines, we define the left derivatives as

$$\left(\frac{\partial J_R}{\partial s_j}\right)^- = \begin{cases} \theta_j'(s_{\max}) + \sum_{b=1}^{B_l} \sum_{i=k_l^b}^{n_l^b} \phi_i' \left(a_{k_l^b} + s_{\text{total}} + \left(i - k_l^b\right) s_{\max}\right) & s_j = \max_{i \neq j} s_i \\ \theta_j'(\sigma_{(l)}) + \sum_{b=1}^{B_l} \sum_{i=k_l^b}^{n_l^b} \left[\phi_i' \left(a_{k_l^b} + s_{\text{total}} + \left(i - k_l^b\right) \sigma_{(l)}\right) \left(1 + i - k_l^b\right)\right] & \sigma_{(l)} = s_j > \max_{i \neq j} s_i \end{cases} \tag{33}$$

for $j = 1, ..., M$.

Since J_R is continuous and convex, yet not everywhere differentiable, we define the subgradients as the left derivative vector ξ with components

$$\xi_j = \left(\frac{\partial J_R}{\partial s_j}\right)^-$$

for all $j = 1, ..., M$. The subgradient directions drive the following descent algorithm with projections, which runs until the stopping condition determined by an ϵ termination tolerance and a d distance metric is satisfied:

Algorithm:

Step 0. Start with an arbitrary initial solution $s^0 = (s_1^0, ..., s_M^0)$
Repeat for $k = 1, 2, ...$

Step 1. Determine the global bottleneck machine $m = \min\{v : s_v^{k-1} = \max_{j=1,...,M} s_j^{k-1}\}$

Step 2. Determine the block structure at the global bottleneck machine m employing Lemma 5

Step 3. Determine ξ_j^{k-1} for all $j = 1, ..., M$

Step 4. Update solution

$$s^k = \Pi\left[s^{k-1} - \eta_k \xi^{k-1}\right] \tag{34}$$

Until $d(s^k, s^{k-1}) < \epsilon$

In Eq. 34, step sizes $\{\eta_k\}_{k=1}^{\infty}$ satisfy the standard conditions

$$\sum_{k=1}^{\infty} \eta_k^2 < \infty, \sum_{k=1}^{\infty} \eta_k = \infty$$

and Π denotes the projection mapping onto the feasible solutions set $\{(s_1, ..., s_M) : s_1 \geq S_1, ..., s_M \geq S_M\}$. Subgradient descent algorithms with projections are known to converge to the optimal solution (see, e.g. in Bertsekas (1995)). The computational complexity per iteration is given as $O(\max(M, N))$, i.e., the computational complexity per iteration is linear in both M and N.

5 Numerical examples

Example 1 We consider the example in Gokbayrak and Selvi (2008), where ten jobs are to be processed in a flow shop of four machines. The total service cost $\theta_j(s_j)$ at machine j is given as

$$\theta_j(s_j) = \frac{\beta_j}{s_j} \tag{35}$$

for some constant β_j, where $\beta = [100, 50, 200, 100]$, while the completion time cost for job C_i is given as

$$\phi_i(x_{i,M}) = 10(x_{i,M} - a_i)^2 \tag{36}$$

where the arrival times of the jobs are $a = [0.0, 2.3, 2.4, 4.9, 5.0, 5.5, 9.0, 9.5, 11.0, 13.0]$. Note that these costs satisfy Assumptions 1 and 2.

The service times are initially set to their lower bounds $s_j^0 = S_j$ for all $j = 1, ..., 4$, where $S = [0.20, 0.20, 0.30, 0.35]$. The termination tolerance ϵ is given to be 10^{-8} and the step size at iteration k is given as $\eta_k = \frac{0.002}{k}$. The implemented distance metric is given as

$$d\left(s^k, s^{k-1}\right) = \max_j \left| s_j^k - s_j^{k-1} \right| \tag{37}$$

The optimization problem R is solved to yield the optimal service times $s^* = [0.4942, 0.3495, 0.5593, 0.4942]$. (The service times in the first 20 iterations are shown in Fig. 1.)

The first machine is a local bottleneck as in all flow shop systems. Operating with the optimal service times s^*, the third machine turns out to be the global bottleneck, and there are no other local bottlenecks. Therefore, the system can be divided into two flushing portions: one is formed of the first and the second machines, and the other one is formed of the third and the fourth machines. From Lemma 1, we expect to see no waiting in front of the second and the fourth machines. Given the arrival times, the minimums of the average interarrival times are evaluated as $\sigma = [\infty, 2.3, 0.1, 1.3, 0.1, 0.3, 1.34, 0.5, 1, 1.\bar{3}]$. From Lemma 3, we expect jobs $\{C_3, C_5, C_6\}$ to wait in front of the first machine, because $\sigma_3 = \sigma_5 \leq \sigma_6 < s_1^*$. They are also expected to wait in front of the third machine, the downstream local (and

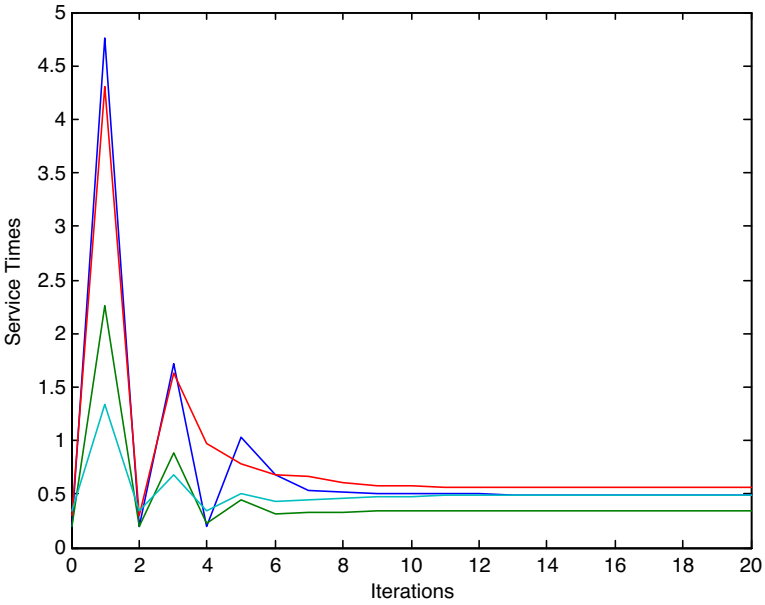

Fig. 1 Evolutions of service times

Table 1 Optimal departure times

Departure times	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8	Job 9	Job 10
Arrival	0.0000	2.3000	2.4000	4.9000	5.0000	5.5000	9.0000	9.5000	11.0000	13.0000
Machine 1	0.4942	2.7942	3.2885	5.3942	5.8885	6.3827	9.4942	9.9942	11.4942	13.4942
Machine 2	0.8437	3.1437	3.6380	5.7437	6.2380	6.7322	9.8437	10.8437	11.8437	13.8437
Machine 3	1.4030	3.7030	4.2623	6.3030	6.8623	7.4216	10.4030	10.9623	12.4030	14.4030
Machine 4	1.8972	4.1972	4.7565	6.7972	7.3565	7.9158	10.8972	11.4565	12.8972	14.8972

global) bottleneck, as stated in Lemma 2. Since $s_1^* \leq \sigma_8 < s_3^*$, job C_8 is expected to wait only in front of the global bottleneck.

The optimal departure times resulting from operating with optimal service times s^* are given in Table 1, where a dark background denotes waiting after departure. It can be verified that the expected waiting characteristics are realized.

In Fig. 1, we observe oscillations during the initial iterations, and afterwards the algorithm enters the "convergence mode". This is very typical behavior for steepest descent methods with decreasing step sizes. Selecting a very small initial step size may eliminate oscillations, however, this selection may also end up with slower convergence. Another factor that affects the performance of the algorithm is the termination tolerance ϵ. Selecting a large ϵ value may result with a "premature termination", i.e., the algorithm may stop far from the optimal. In short, the selection of ϵ and η_k affects the performance. In the following example, we fix the selection over several problems and observe the results.

Example 2 In this example, we compare the performance of the subgradient descent algorithm solving R against *cvx*, a modeling system for convex programming developed in Stanford University (see in Grant and Boyd 2007), solving Q under different M and N settings. The computation environment is Matlab running on a dual core 2.0 GHz PC with 2 GB memory. The comparisons are based on averages over ten optimization problems (obtained by varying arrival sequences a and the cost parameter vector β) for each M and N combination. The distance measure in Eq. 37 is employed with an ϵ value of 10^{-5}, and the step sizes are given by $\eta_k = \frac{10^{-5}}{k}$.

For all M and N combinations, the subgradient descent algorithm solving R produced the same solutions (according to our precision determined by the ϵ value) as the cvx solver solving Q. Moreover, our subgradient descent algorithm not only improved the solution times but also increased the solvable system sizes as can be observed in Table 2. Note that a dash sign indicates an "out of memory" crash.

Table 2 Average CPU times in seconds

N	$M = 20$		$M = 40$		$M = 60$	
	Q	R	Q	R	Q	R
500	8.06	0.76	11.68	1.13	61.09	1.74
1,000	26.54	1.73	48.33	2.98	355.05	4.38
1,500	51.37	3.28	99.50	5.46	2,251.50	8.17
2,000	82.51	5.40	169.32	9.07	–	12.90
2,500	130.39	7.79	–	12.64	–	18.47
3,000	–	10.80	–	17.81	–	25.68

6 Conclusion

This paper considered manufacturing flow shops formed of traditional non-CNC machines processing identical jobs. Unlike computer controlled machines that can modify service without a setup, traditional machines require a human operator to turn several knobs for service time modifications. The mode of operation during mass production is to set the service times initially to a good value so as not to have the production line stop for frequent setups. This mode also eliminates human errors. The resulting system is modeled as an initially controllable deterministic flow shop system, for which we reported a convex optimization problem formulation Q in Gokbayrak and Selvi (2008). Since that formulation required a convex programming problem solver, which may not be available in smaller manufacturing companies, and needed several GB's of memory for large problems, we proposed an alternative formulation and a subgradient descent solution method employing subgradients for directions. For this formulation, some waiting and completion time characteristics of fixed service time flow shop systems were derived and exploited. As demonstrated by the numerical examples, substantial improvements in solution times and solvable system sizes were observed.

For the same flow shop systems, one may lower the cost by infrequent setups, which both incur costs and consume time. The problem with setups is the topic of ongoing research.

References

Bertsekas DP (1995) Nonlinear programming. Athena Scientific, Belmont, Massachusetts

Cassandras CG, Lafortune S (1999) Introduction to discrete event systems. Kluwer Academic, Dordrecht

Cassandras CG, Pepyne DL, Wardi Y (2001) Optimal control of a class of hybrid systems. IEEE Trans Automat Control 46(3):398–415

Cassandras CG, Liu Q, Pepyne DL, Gokbayrak K (1999) Optimal control of a two-stage hybrid manufacturing system model. In: Proceedings of 38th IEEE conference on decision and control, pp 450–455

Cho YC, Cassandras CG, Pepyne DL (2001) Forward decomposition algorithms for optimal control of a class of hybrid systems. Int J Robust Nonlinear Control 11:497–513

Gazarik M, Wardi Y (1998) Optimal release times in a single server: an optimal control perspective. IEEE Trans Automat Control 43(7):998–1002

Gokbayrak K, Cassandras CG (2000) Constrained optimal control for multistage hybrid manufacturing system models. In: Proceedings of 8th IEEE Mediterranean conference on new directions in control and automation

Gokbayrak K, Selvi O (2006) Optimal hybrid control of a two-stage manufacturing system. In: Proceedings of ACC, pp 3364–3369

Gokbayrak K, Selvi O (2007) Constrained optimal hybrid control of a flow shop system. IEEE Trans Automat Control 52–12:2270–2281

Gokbayrak K, Selvi O (2008) Optimization of a flow shop system of initially controllable machines. IEEE Trans Automat Control 53:2665–2668

Grant M, Boyd S (2007) CVX: Matlab software for disciplined convex programming. http://stanford.edu/~boyd/cvx

Kalpakjian S, Schmid SR (2006) Manufacturing engineering and technology. Pearson Prentice Hall, Singapore

Moon J, Wardi Y (2005) Optimal control of processing times in single-stage discrete event dynamic systems with blocking. IEEE Trans Automat Control 50(6):880–884

Pepyne DL, Cassandras CG (1998) Modeling, analysis, and optimal control of a class of hybrid systems. J Discrete Event Dyn Syst: Theory Appl 8(2):175–201

Pepyne DL, Cassandras CG (2000) Optimal control of hybrid systems in manufacturing. Proc IEEE 88(7):1108–1123
Pinedo M (2002) Scheduling: theory, algorithms, and systems. Prentice Hall, Englewood Cliffs
Wardi Y, Cassandras CG, Pepyne DL (2001) A backward algorithm for computing optimal controls for single-stage hybrid manufacturing systems. Int J Product Res 39–2:369–393
Zhang P, Cassandras CG (2002) An improved forward algorithm for optimal control of a class of hybrid systems. IEEE Trans Automat Control 47–10:1735–1739

Kagan Gokbayrak was born in Istanbul, Turkey, in 1972. He received the B.S. degrees in mathematics and in electrical engineering from Bogazici University, Istanbul, the M.S. degree in electrical and computer engineering from the University of Massachusetts, Amherst, and the Ph.D. degree in manufacturing engineering from Boston University, Boston, MA, in 1995, 1995, 1997, and 2001, respectively. From 2001 to 2003, he was a Network Planning Engineer at Genuity, Inc., Burlington, MA. Since 2003, he has been a faculty member in the Industrial Engineering Department of Bilkent University, Ankara, Turkey. He specializes in the areas of discrete-event and hybrid systems, stochastic optimization, and computer simulation, with applications to inventory and manufacturing systems.

Omer Selvi was born in Nigde, Turkey, in 1976. He received the B.S., M.S., and Ph.D. degrees in industrial engineering from Bilkent University, Ankara, Turkey, in 1999, 2002, and 2008, respectively. His research interests are in the fields of discrete event systems and stochastic optimization. He is currently serving in the Turkish Armed Forces.