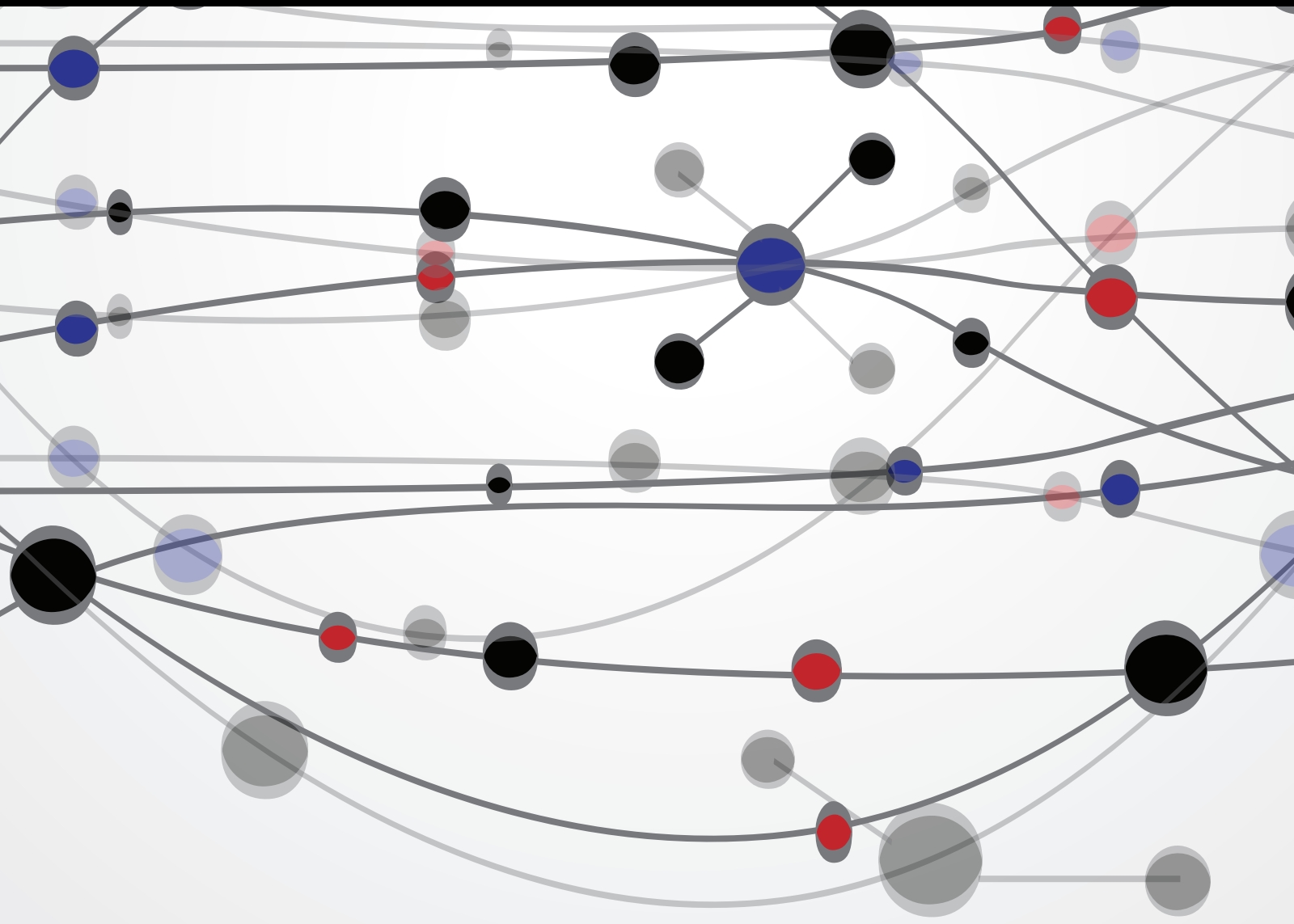
The Scientific World Journal

New Developments in Scheduling Applications

Guest Editors: M. Duran Toksari, Daniel Oron, Emel Kizlkaya Aydogan, and Jorge G. Barbosa



New Developments in Scheduling Applications

The Scientific World Journal

New Developments in Scheduling Applications

Guest Editors: M. Duran Toksari, Daniel Oron,
Emel Kizlkaya Aydogan, and Jorge G. Barbosa

Copyright © 2014 Hindawi Publishing Corporation. All rights reserved.

This is a special issue published in "The Scientific World Journal." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Contents

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 268941, 2 pages
http://dx.doi.org/10.1155/2014/268941

Editorial
New Developments in Scheduling Applications

M. Duran Toksarı,[1] Daniel Oron,[2] Emel Kızılkaya Aydoğan,[1] and Jorge G. Barbosa[3]

[1] *Department of Industrial Engineering, Erciyes University, 38039 Kayseri, Turkey*
[2] *Business School, The University of Sydney, NSW 2006, Australia*
[3] *Universidade do Porto, Faculdade de Engenharia, Departamento de Engenharia Informática, Laboratório de Inteligência Artificial e Ciência dos Computadores, Rua Dr. Roberto Frias, 4200-465, Portugal*

Correspondence should be addressed to M. Duran Toksarı; dtoksari@erciyes.edu.tr

Received 17 March 2014; Accepted 17 March 2014; Published 28 April 2014

Copyright © 2014 M. Duran Toksarı et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Scheduling problems are widely recognized as important optimization problems. Thus, scheduling theory has become a fundamental area within the general field of combinatorial optimization. Multiprocessor and shop scheduling problems are known to be hard to solve optimally. Scheduling problems are naturally very varied, both in application domains and in featured constraints. To date, researchers have been working on scheduling problems derived from new applications such as scheduling problems in logistical airport operations management processes, decentralized systems and selfish organizations, grid computing, and bioinformatics. These scheduling problems reflect real-life situations for including learning effects or deteriorating jobs.

In the call for papers, we advised that all authors should be encouraged to focus on scheduling heuristics, scheduling with some effects such as learning and deterioration, grouping and sequencing operations in multistage systems, scheduling in flexible shops, scheduling under some constraints such as precedence, batching/lot sizing, setups, and further technologies, scheduling under uncertainty, and scheduling in a supply chain. The obtained responses gratified us with a total of 24 submissions. All of them were peer reviewed according to high standards of this journal. At the end of the process, we accepted seven papers. The accepted papers represent excellent work that spans across a wide variety of cutting edge scheduling problems and applications.

D. C. Dietz proposes a study on computing the expected cost of an appointment schedule for statistically identical customers with probabilistic service times. In this paper, the author presented a cogent method to compute the expected cost of an appointment schedule. Customers are statistically identical, the service time distribution has been known as mean and variance, and no-shows occur with time-dependent probability.

X. Shi and D. Xu develop a solution for single-machine scheduling with increasing linear maintenance durations by the best possible approximation algorithms. They used the linear form $f(t) = a + bt$ with $a \geq 0$ and $b > 1$. They proposed an approximation algorithm named FFD-LS2I with a worst-case bound of 2 for problem. Furthermore, they also showed that there is no polynomial time approximation algorithm with a worst-case bound for the problem with $b \geq 0$ unless $P = NP$, which implies that the FFD-LS2I algorithm is the best possible algorithm for the case $b > 1$.

T. Luo and Y. Xu investigate semionline scheduling on two machines with grade of service (GoS) levels and partial information of processing time. They worked on three different semionline versions (knowing total processing time of the jobs with higher GoS level or knowing total processing time of the jobs with lower GoS level or knowing both in advance) and proposed algorithms with competitive ratios to solve these semionline versions.

W. Liu et al. develop a time scheduling model of logistics service supply chain (LSSC) based on the customer order decoupling point. They tested their algorithm using numerical analysis for a specific example and obtained interesting results. The order completion time of the LSSC can be delayed or be ahead of schedule but cannot be infinitely advanced or infinitely delayed. The optimal comprehensive performance can be effective if the expected order completion time is appropriately delayed.

F. Tahriri et al. propose a study on a fuzzy mixed model assembly line sequencing and scheduling using multiobjective dynamic fuzzy genetic algorithm. They proposed a new multiobjective dynamic fuzzy genetic algorithm to solve a fuzzy mixed-model assembly line sequencing problem where the primary goal is to minimize the makespan, setup time, and cost simultaneously. They performed a simulation to compare the proposed novel optimization algorithm and the standard genetic algorithm in mixed assembly line sequencing model. The obtained results highlight that the performance and effectiveness of the proposed novel optimization algorithm are more efficient than the performance of the standard genetic algorithm.

V. Fernandez-Viagas and J. M. Firaminan study integrated project scheduling and staff assignment with controllable processing times. They proposed an integer programming model to solve problem, together with some extensions to cope with different settings. Furthermore, the advantages of the controllable processing times approach are compared with the fixed processing times, and they applied a simple GRASP algorithm due to the complexity of the integrated model.

H. Gong et al. present a parallel-batch scheduling and transportation coordination with waiting time constraints. They solved the parallel-batch scheduling problem that incorporates transportation of raw materials or semifinished products before processing under the consideration of waiting time constraints. Furthermore, they proposed an optimal algorithm in polynomial time to solve the case with equal processing times and equal transportation times for each order.

In summary, the seven papers represent some of the latest and most promising research results on scheduling problems. We believe that they make significant impact on solving both theoretical problems and real life applications. We are confident that this special issue will stimulate further research in this area.

Acknowledgment

M. Duran Toksarı's research was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK).

M. Duran Toksarı
Daniel Oron
Emel Kızılkaya Aydoğan
Jorge G. Barbosa

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 924120, 16 pages
http://dx.doi.org/10.1155/2014/924120

Research Article

Integrated Project Scheduling and Staff Assignment with Controllable Processing Times

Victor Fernandez-Viagas and Jose M. Framinan

Industrial Management, School of Engineering, University of Seville, Camino de los Descubrimientos, s/n, 41092 Seville, Spain

Correspondence should be addressed to Victor Fernandez-Viagas; vfernandezviagas@us.es

Received 11 January 2014; Accepted 19 February 2014; Published 24 April 2014

Academic Editors: J. G. Barbosa and D. Oron

Copyright © 2014 V. Fernandez-Viagas and J. M. Framinan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper addresses a decision problem related to simultaneously scheduling the tasks in a project and assigning the staff to these tasks, taking into account that a task can be performed only by employees with certain skills, and that the length of each task depends on the number of employees assigned. This type of problems usually appears in service companies, where both tasks scheduling and staff assignment are closely related. An integer programming model for the problem is proposed, together with some extensions to cope with different situations. Additionally, the advantages of the controllable processing times approach are compared with the fixed processing times. Due to the complexity of the integrated model, a simple GRASP algorithm is implemented in order to obtain good, approximate solutions in short computation times.

1. Introduction

The lead time required to carry out a project—project makespan—has turned to be one of the main sources of competitive advantage for companies (see, e.g., [1, 2]). Since, in many cases, the processing times of the tasks that compose a project depend on the resources allocated to the task (see, e.g., [3]), it is clear that the project lead time depends both on the scheduling of the tasks and on the allocation of staff to these tasks. Regarding task scheduling, it encompasses the sequence to be followed by each task over time, and it is usually regarded as an important aspect of the industrial organization [4] and production [5]. On the other hand, staff assignment is essential in a project [6] and it can be defined as the allocation of company workers to different tasks. Traditionally, staff assignment is included within the manpower planning as one of its steps. Thus, several authors [7–10] have considered the following structure: (1) planning; (2) scheduling; (3) allocation. In staff assignment, worker's satisfaction tends to be maximized whereas makespan and production costs are often minimized in task scheduling. To execute a task, it is essential that a particular employee is available at that time; otherwise the realization of such task should be postponed until the employee becomes available and, consequently, staff assignment directly influences task schedule. Due to the importance of the scheduling and the assignment in a project environment and since both are very related, it is critical to address both problems in an integrated manner. Furthermore, the majority of scheduling problems in the literature consider fixed processing times [11] and most of the rest assume that processing times depend linearly on the amount of resources assigned [12]. As lead times strongly depend on the processing times of the task, and since the processing time of a task depends on the number of employees assigned to this task, in this paper we explicitly take into account this relationship.

In this paper, an integration of project scheduling and staff assignment (PSSA) problem with controllable processing times (CPT) is presented using renewable resources and nonpreemptive tasks. Minimization of the makespan has been chosen as the objective function since delays can lead to an increase in costs and even to the loss of customers. This problem can be placed within the project management process. According to Demeulemeester and Herroelen [13], the project life cycle is as follows.

Table 1: Names for the integrated problem.

Reference	Name used for the integrated problem
Bellenguez and Néron [14], Bellenguez-Morineau and Néron [15]	Multiskill project scheduling problem
Brucker and Knust [16], Drezet and Billaut [17]	Project scheduling with labor constraints
Vairaktarakis [18]	Resource-constrained job assignment problem
Valls et al. [19]	Skilled workforce project scheduling
Corominas et al. [20]	Problem of assigning and scheduling a set of tasks to a set of workers
Drexl [21]	Scheduling of project networks by job assignment
Dodin and Elimam [22]	Problem of audit staff scheduling
Heimerl and Kolisch [23], Kolisch and Heimerl [24], Gutjahr et al. [25], and Gutjahr et al. [26]	Project scheduling and staffing
Wu and Sun [27]	Project scheduling and staff assignment

(i) *Concept Phase*. There is a need of a customer to perform a project. This need is transmitted to the company.

(ii) *Definition Phase*. First, the goal of the project is defined. Next, the work content is defined and, finally, a project strategy is elaborated to achieve the goal.

(iii) *Planning Phase*. The project is divided into tasks. Then, their processing times are estimated and both the resources requirements and the precedence relationships between tasks are identified.

(iv) *Scheduling Phase*. Both task scheduling and the amount of resources in each period of time are defined in this phase

(v) *Control Phase*. In this phase it is controlled that the project is implemented following the aspects defined in the planning and scheduling phases.

(vi) *Termination Phase*. This phase corresponds to the delivery of the results of the project.

Note that between the scheduling phase (where the amount of resources is known) and the control phase there must be an assignment of resources to the tasks if the resources are not identical. For these cases, an additional phase is therefore needed. This phase—named "resources assignment phase"—must be placed between the scheduling phase and the control phase. Alternatively, integration between the scheduling and the resources assignment phases can be done. The latter is the adopted approach in this paper, which is organized as follows: a state of the art and a description of the problem are shown in Section 2. An extended explanation of the CPT is also presented, while the problem statement and a formulation of the integer linear programming model are shown in Section 3. Additionally this integer linear programming model is compared with a multimode formulation. In Section 4, we define a simple GRASP heuristic algorithm for the problem. Section 5 contains computational experiments based on a test bed. Besides, a comparison between the model with and without CPT is shown there. Lastly, conclusions are described in Section 6.

2. Literature Review

The model presented in this paper includes two decision problems: task scheduling and staff assignment. Task scheduling is traditionally denoted in the literature as *project scheduling problem* (PSP). Furthermore, if limitations regarding the number of resources per activity are assumed, it is named *resource-constraint project scheduling problem* (RCPSP). This problem has been extensively studied in the literature, and recent reviews can be found in Węglarz et al. [32] and Hartmann and Briskorn [33]. Staff assignment is a type of assignment decision problem that has also been extensively addressed [34]. Nevertheless, the integration between project scheduling and staff assignment has not been so comprehensively studied, and hence there is still no consensus about the name of this joint problem (see in Table 1 different names that have been used to denote the problem). In our work, we consider the name "project scheduling and staff assignment" as denomination for the problem.

Our problem is related to several contributions in the literature, which are summarized in Table 2. Bassett [29] presents a model of project scheduling and staff assignment considering time windows for the completion times of the tasks. Vairaktarakis [18] adds precedence relationships to the integrated problem. However, no time units are used in the linear programming model (the order of the tasks defines the schedule of the project). Time units are included in a similar problem by Bellenguez and Néron [14] and Bellenguez-Morineau and Néron [15] where some lower bounds are calculated in the former reference, while in the latter the authors implement a branch and bound algorithm. External and internal resources constraints are considered by Kolisch and Heimerl [24]. Wu and Sun [27] present the integrated problem proposing a nonlinear model, which is solved by a genetic algorithm. Although learning effect (i.e., the longer an employee works on a task, the greater his/her efficiency is) is considered in the model, neither precedence relationships nor skill constraints are used in their model. Gutjahr et al. [26] consider portfolio selection and skills in the integrated problem with learning effect but with precedence relationship which are included by Gutjahr et al. [25]. Precedence relationships are also included by Corominas et al. [20] in an

Table 2: Summary of related papers.

	Project sched.	Staff assign.	Prec. relations	Release time	Due dates	Preemp.-nonPreemp.	Learning effect	CPT_{Skill}	$CPT_{N^{\circ}\ of\ workers}$	Move const.	Outsourcing staff	Project selection	Resources per task	Skill	Objective function	Exact methods and solvers	Aprox. alg.
Alfares and Bailey [28]	X	Global	X	X		NP			Data[1]						Cost	[2]	Heuristic
Bassett [29]	X	X			X	NP							1	X	Cost	CPLEX	Heuristic
Bellenguez and Néron [14]	X	X	X			NP							n	X	Makespan	Branch and bound	
Bellenguez-Morineau and Néron [15]	X	X	X			NP							n	X	Makespan	Lower bound	
Corominas et al. [20]	X	X	X	X		P	X						1		Makespan	CPLEX	
Dodin and Elimam [22]	X	X	X			NP		Data[1]					1		Cost	Lindo	
Drexl [21]	X	X	X			NP		Data[1]					1		Cost	Branch and bound	Monte Carlo
Drezet and Billaut [17]	X	X	X	X	X	NP				X			n	X	Lateness	CPLEX	Tabu search
Gutjahr et al. [25]	X	X	X	X	X	P	X					X	n	X	Multiobjective	CPLEX	Greedy procedure
Gutjahr et al. [26]	X	X		X	X	P	X					X	n	X	Multiobjective	CPLEX	NSGA-II and P-ACO
Hachicha et al. [30]		X						Data[1]							Cost and preference	Lingo	
Heimerl and Kolisch [31]		X					X	Non Linear						X	Cost		COIN-OR's Ipopt
Kolisch and Heimerl [24]	X	X	X	X	X	NP					X		n	X	Cost	CPLEX	Genetic alg. and tabu search
Valls et al. [19]	X	X	X	X	X	NP		Linear					1	X	Multiobjective		Genetic alg.
Vairaktarakis [18]	X	X	X										1	X	Makespan	Lower bound	Heuristics
Wu and Sun [27]	X	X			X	P	X				X		1		Cost	CPLEX	Genetic alg.
This paper	X	X	X	X		NP			Piecewise linear	X			n	X	Makespan	CPLEX Gurobi	GRASP

[1] Processing times are preprocessed depending on the number of assigned employees or on the level of skills of the assigned employee(s) and then they are introduced in the model as data.
[2] An integer programming model is solved, although the way to do it is not explained.

integrated problem, similar to the problem of Wu and Sun [27], also with learning effect. Drezet and Billaut [17] propose a linear programming model for a project scheduling and staff assignment problem. However, they do not consider learning effects in the model and include move constraints (i.e., the number of assignment movements of the employees from one nonfinished task to another is bounded). Moreover, the number of employees assigned to each task at each period of time is set between a minimum and a maximum depending on the level of skills of the workers. However, the processing times of the tasks are assumed to be fixed and not depending on the number of employees assigned. Since no solutions can be found for medium-sized instances of the problem, a tabu search algorithm was implemented using a greedy algorithm as initial solution.

A common feature in all the aforementioned works is that none of them include variable processing times. Regarding CPT, Hachicha et al. [30] propose a staff assignment model considering multiskilled employees, preference of the employees for some tasks, and skill-based processing times (i.e., the processing times of the tasks depend on the skills of the employee). Nevertheless, they are preprocessed before solving the model, which means that the processing times do not change during the planning horizon. More specifically, processing times are first calculated depending on the skills of each employee and then introduced in the model as data. Heimerl and Kolisch [31] analyze a staff assignment problem without scheduling considering learning effect on the employees, so the processing times are now variables of the problem as they depend on the experience of the staff. In order to solve this problem, a nonlinear model is implemented. Hachicha et al. [30] and Heimerl and Kolisch [31] use CPT in their models, although none of them solve the integrated problem (only the staff assignment problem). Variable processing times in a scheduling problem can be found in Alfares and Bailey [28]. They use a linear programming model to schedule the task of a project and to assign the number of employees to each task; that is, the variables of the problem are the starting times of each task, the processing times, and the number of employees per period. Regarding the processing times, they are chosen to minimize the costs, but they do not depend on any variable. Moreover, staff assignment for each employee is not considered there.

CPT in an integrated PSSA problem are considered in Drexl [21], Dodin and Elimam [22] with the objective of costs minimization. However, in their models, the processing times depend on the skills of the employee assigned (and consequently are introduced in the model as data), but not on the amount of resources allocated; that is, processing times are defined by a matrix of two dimensions where the rows are the processing times of the tasks and the columns the employees. In Valls et al. [19], each task has to be performed by a single employee and its processing time increases or decreases depending on the efficiency of the employee. Nonpreemptive tasks with release times and due dates are taken into account, as well as precedence relations with time lags. No programming model is implemented, but a hybrid genetic algorithm is developed to solve the cost minimization problem. This contribution is the one most related to our problem, since the authors implement an integrated PSSA with CPT. However, in our paper, a linear programming model is considered to minimize the makespan of the project where the processing times of the tasks depend piecewise linearly on the number of employees assigned instead of the efficiency level of the employee and multiple assignments to each task are also allowed as well, while in Valls et al. [19] each task must be implemented by a single employee. To the best of our knowledge, the proposed model has not been analyzed before.

3. Model Proposal for the Integrated PSSA Problem

3.1. Problem Statement. This paper presents a PSSA problem in which a company is responsible for the implementation of a project consisting of J tasks (with task $j = 1, \ldots, J$) with (known) precedence relations between tasks (i.e., some tasks should be finished before carrying out others). The order of execution of each task has to be decided to minimize the corresponding objective function. Each task must be performed by some workers from total of E employees (employee $e = 1, \ldots, E$) with certain skills in each t period ($t = 1, \ldots, T$) of the planning horizon T.

It is assumed that each task j has a release time r_j, so it must always start after this time. Furthermore, once a task starts, it runs continuously until its end (nonpreemptive approach). One employee can be assigned to a task if the employee possesses the required skills for the task. We assume that there is an optimal number of employees, R_j, to be assigned to each task, although overcoverage and undercoverage of employees are allowed by the model. Under- and overcoverage would lead to different values of the efficiency of the employees and, consequently, to different processing times of the tasks.

3.2. Controllable Processing Times (CPT). In this paper, we assume that processing times depend on the resources (or employees, as we would use both terms as equivalent in this paper) assigned. Such type of processing times has been studied in the literature as a function of the resources allocated and of the experience of the workers (level of skills). The relationship between time and amount of resources has been mainly analysed under two different approaches, namely, linear and convex (see the review by [12]). In the linear approach (see, e.g., [35]), a linear relationship between the processing times of a task and the resources allocated to this task is assumed. Denoting p_j as the processing time of task j and u_j as the number of resources allocated to this task, this relationship can be expressed as $p_j = a - b \cdot u_j$, with a and b constants. However, this approach is not well suited to many realistic situations, since according to Belbin [36], there must be R_j an optimal number of employees to be assigned to task j in order to achieve maximum workers'

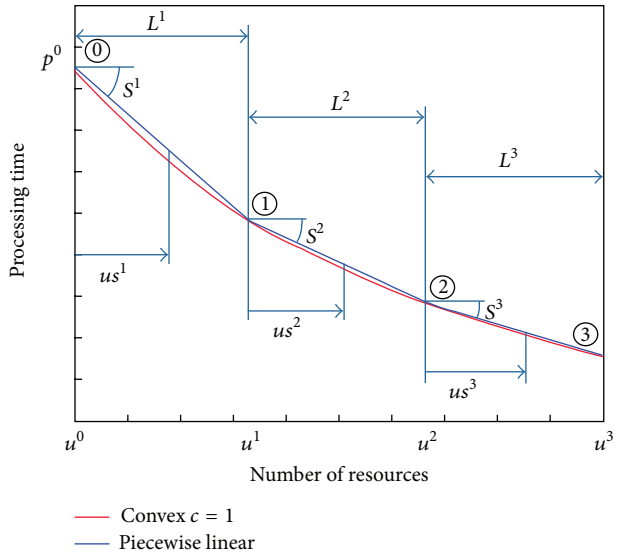


FIGURE 1: Piecewise linear relationship and convex relationship with $c = 1$. Subscripts are removed for simplicity in the figure.

efficiency (point $[R_j, pd_j]$). Therefore, it is easy to see that the linear approach is not flexible to this concept, at least for values to each side of the efficiency level since the efficiency point of worker must always be in an extreme of the line (see, e.g., Figure 2(a), where the efficiency point is placed for a/b resources).

In the convex approach it is assumed that there is an inverse relationship between processing times and employees (see, e.g., [37]), where c and W_j (normally denoted workload of the task j) are constants:

$$p_j = \left(\frac{W_j}{u_j}\right)^c. \tag{1}$$

For $c = 1$, the number of resources and the processing times are inversely proportional (e.g., doubling the employees would cut the processing time by half and vice versa; see the red line of Figure 1). The convex relationship is closer to real life than the linear and it has been used for many actual government and industrial projects [38], distributed communication network, time-sharing computing system, and chemical plant or commercial construction projects [39]. Nevertheless, its nonlinearity would preclude modeling the problem using linear programming. Thus, in this paper a piecewise linear relationship to represent the relationship between processing times and amount of resources assigned to the task is proposed. The relationship is shown in (2), S_j^i being the slope of each section for a task j in the piecewise linear relationship and us_j^i the number of employees fulfilled of that task in each section, where section i has a length (L_j^i). This piecewise linear relationship can be adjusted to the convex relationship (with $c = 1$) as shown in the blue line of Figure 1. This relationship is over the convex relationship representing a safer configuration of processing times and amount of resources:

$$\begin{aligned} p_j &= p_j^0 + S_j^1 \cdot us_j^1 + \cdots + S_j^k \cdot us_j^k, \\ \text{with } us_j^i &= L_j^i \quad \forall i < \frac{m}{0} < us_j^m < L_j^i, \\ S_j^i &= \frac{p_j^i - p_j^{i-1}}{u_j^i - u_j^{i-1}}. \end{aligned} \tag{2}$$

The superscript for processing times, p_j^i, and for the amount of resources, u_j^i, denotes that they correspond to point i.

According to Lanigan [40], the convex relationships between the processing time of a task and the number of employees assigned do not entirely match the reality, and there shall be a penalty due to assigning different numbers of employees. On the one hand, if $u_j > R_j$, then a penalty for communication exists. In contrast (see, e.g., [41, 42]), if $u_j < R_j$ then there is a penalty for lack of specialization [43]. Hence, each feasible point different than the optimum (R_j, pd_j) must be placed over the convex curve; otherwise many optima would exist and there would be no penalties for under- and overcoverage. The feasible area formed by these points is represented by the light red area in Figure 2(a) (this feasible area is also taken into account by [44, 45]). Moreover, it shall be considered that the processing time of a task decreases when the number of employees increases; otherwise it would make no sense to assign more employees to the tasks. Thus, the feasible area can be reduced by considering that the processing time of a task cannot increase when the number of employees increases (Figure 2(b)). In this way, the feasible relationships between both aspects must be in this area. Therefore, we propose using a piecewise linear relationship with a maximum and minimum possible number of workers on each task. Depending on the maximum and minimum possible number of workers, the slopes of the lines must be different to avoid that both lines are under the convex curve (i.e., in the infeasible region) in any point. The directions of the slope of the piecewise relationships are placed in the feasible region and shown in Figure 2(c). By doing so, linear programming can be used to model and solve the problem. Furthermore, the piecewise relationship is on the safe side with respect to the convex relation, since the piecewise relationship is always over the convex relation.

To define the slopes of the lines, we propose introducing the parameters kr and kl. The higher these parameters are, the bigger the penalties for under- and overassignment of employees with respect to the optimal value, and consequently the closer the problem to the PSSA with fixed processing times (FPT). In the most extreme case, both problems are the same. The piecewise linear relation left and right of the optimum can be written as follows (see Figure 2(d)):

$$\begin{aligned} \text{Left: } p_j &= pd_j \cdot \left(1 + kl \cdot \frac{R_j - u_j}{R_j}\right) \\ \text{Right: } p_j &= pd_j \cdot \left(1 - \frac{u_j - R_j}{kr \cdot R_j}\right). \end{aligned} \tag{3}$$

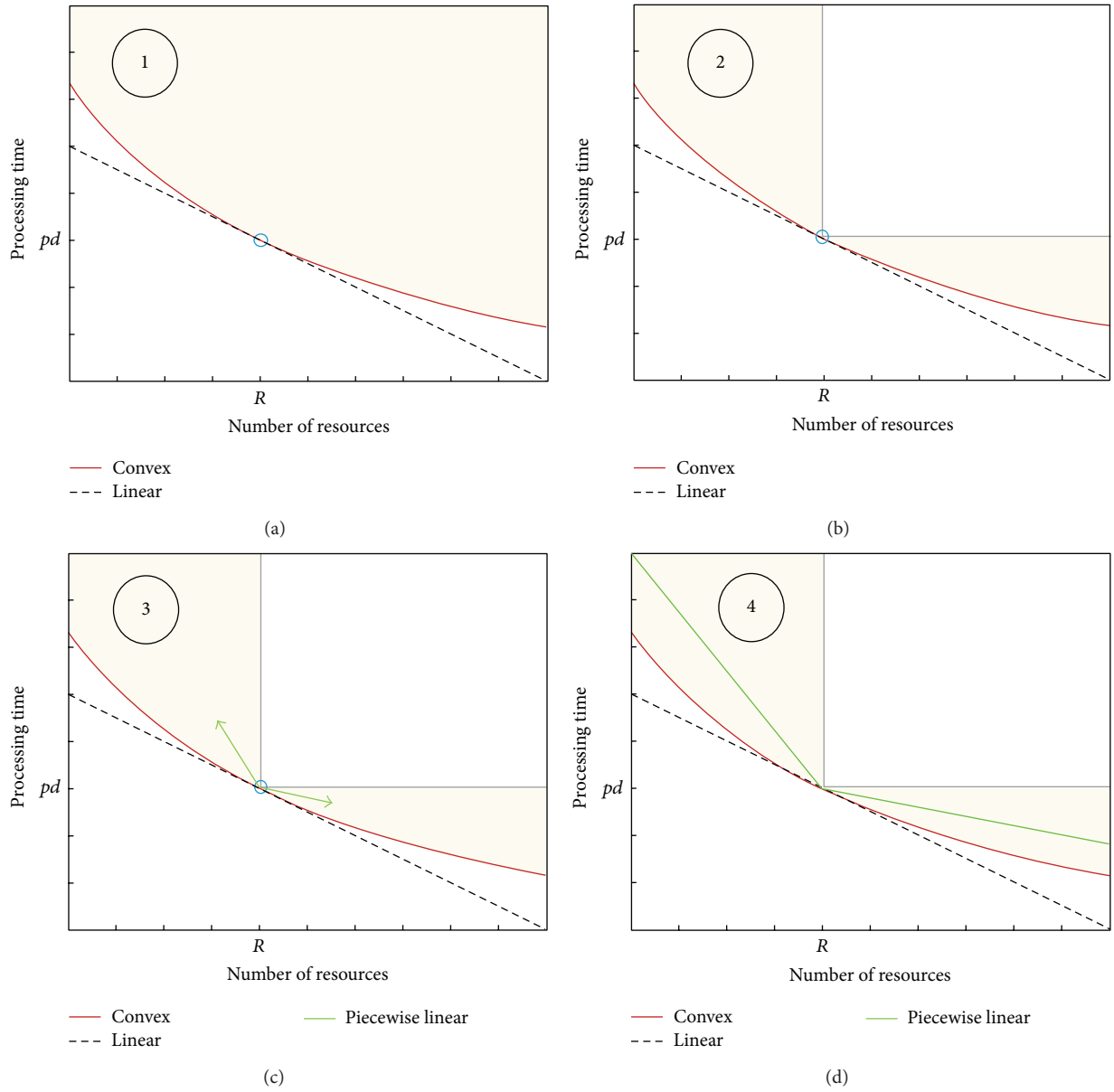


Figure 2: Relation between processing times and number of employees (shaded area).

Analogously, let us now consider $R_j - u_j$ as h_j^+ the undercoverage of task j and $u_j - R_j$ as the overcoverage of task j, h_j^-:

$$\text{Left: } p_j = pd_j \cdot \left(1 + kl \cdot \frac{h_j^+}{R_j}\right)$$
$$\text{Right: } p_j = pd_j \cdot \left(1 - \frac{h_j^-}{kr \cdot R_j}\right). \quad (4)$$

3.3. Formulation of the Model

3.3.1. Proposed Integer Linear Programming Model. In the previous section, we have presented a mechanism for formulating a realistic relationship between the processing times of the task and the number of employees assigned. Furthermore, this relationship can be embedded into a piecewise linear function; therefore a linear programming model can be formulated for the PSSA. In this section, a formal description of this model is given.

Data

r_j: earliest starting time (release time) of task j.

b_{ej} $\begin{cases} 1 \text{ if the employee } e \text{ can perform task } j \\ 0 \text{ otherwise.} \end{cases}$

pr_{ij} $\begin{cases} 1 \text{ if task } i \text{ precedes task } j \\ 0 \text{ otherwise.} \end{cases}$

R_j: optimal number of employees of task j.

pd_j: processing time of task j for the optimal number employees R_j.

LV_j: maximum allowed undercoverage for task j.

UV_j: maximum allowed overcoverage for task j.

kd, kn: constants related to the slopes of the lines defining the processing times.

Variables

x_{jt} $\begin{cases} 1 \text{ if task } j \text{ starts in period } t \\ 0 \text{ otherwise.} \end{cases}$

y_{ejt} $\begin{cases} 1 \text{ if employee } e \text{ performs task } j \text{ during period } t \\ 0 \text{ otherwise.} \end{cases}$

h_j^+: undercoverage in task j.

h_j^-: overcoverage in task j.

p_j: processing time of task j.

Auxiliary Variables

z_{jt} $\begin{cases} 1 \text{ if task } j \text{ is being processed at the begin of period } t \\ 0 \text{ otherwise.} \end{cases}$

st_j: starting time of task j.

ct_j: completion time of task j.

$C_{\max}$: project makespan.

ah_{jt}: auxiliary variable. It is used to enforce that the constraint set (18) is satisfied if z_{jt} is zero since the auxiliary variable takes the value of the over- or undercoverage.

Model

$$\text{Min } C_{\max} \tag{5}$$

$$\sum_{t=1}^{T} x_{jt} = 1 \quad \forall j = 1, \ldots, J \tag{6}$$

$$st_j = \sum_{t=1}^{T} x_{jt} \cdot t \quad \forall j = 1, \ldots, J \tag{7}$$

$$ct_j = st_j + p_j - 1 \quad \forall j = 1, \ldots, J \tag{8}$$

$$st_j \geq r_j \quad \forall j = 1, \ldots, J \tag{9}$$

$$C_{\max} \geq ct_j \quad \forall j = 1, \ldots, J \tag{10}$$

$$y_{ejt} \leq b_{ej} \cdot z_{jt} \quad \forall e = 1, \ldots, E, \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{11}$$

$$st_j - st_i \geq p_i \cdot pr_{ij} - N \cdot \left(1 - pr_{ij}\right) \quad \forall i, j \tag{12}$$

$$\sum_{e=1}^{E} y_{ejt} + ah_{jt} + \left(h_j^+ - h_j^-\right) = R_j \cdot z_{jt} \quad \forall j = 1, \ldots, J, \quad \forall t = 1, \ldots, T \tag{13}$$

$$ah_{jt} \leq \left(1 - z_{jt}\right) \cdot N \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{14}$$

$$ah_{jt} \geq -\left(1 - z_{jt}\right) \cdot N \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{15}$$

$$h_j^+ \leq LV_j \quad \forall j = 1, \ldots, J \tag{16}$$

$$h_j^- \leq UV_j \quad \forall j = 1, \ldots, J \tag{17}$$

$$\sum_{j=1}^{J} y_{ejt} \leq 1 \quad \forall e = 1, \ldots, E, \forall t = 1, \ldots, T \tag{18}$$

$$p_j \leq \frac{1}{2} + pd_j \cdot \left(1 - \frac{h_j^-}{kr \cdot R_j} + kl \cdot \frac{h_j^+}{R_j}\right) \quad \forall j = 1, \ldots, J \tag{19}$$

$$p_j > -\frac{1}{2} + pd_j \cdot \left(1 - \frac{h_j^-}{kr \cdot R_j} + kl \cdot \frac{h_j^+}{R_j}\right) \quad \forall j = 1, \ldots, J \tag{20}$$

$$\sum_{t=1}^{T} z_{jt} = p_j \quad \forall j = 1, \ldots, J \tag{21}$$

$$z_{jt} \geq x_{jt} \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{22}$$

$$1 - z_{jt} \geq \frac{t - ct_j}{T} \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{23}$$

$$1 - z_{jt} \geq \frac{st_j - t}{T} \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{24}$$

$$z_{jt}, x_{jt} \in \{0, 1\}, \quad ah_{jt} \text{ free} \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{25}$$

$$y_{ejt} \in \{0, 1\} \quad \forall e = 1, \ldots, E, \forall j = 1, \ldots, J, \forall t = 1, \ldots, T \tag{26}$$

$$ct_j, st_j, h_j^+, h_j^-, p_j \geq 0 \quad \forall j = 1, \ldots, J \tag{27}$$

$$C_{\max} \geq 0. \tag{28}$$

Objective function (5) minimizes the project makespan. Constraint set (6) guarantees that each task must start exactly once. Constraint sets (7) and (8) define the starting and completion times of each task, respectively. Constraint set (9) assures that a task cannot start before its release time. Constraint set (10) serves to obtain the makespan. Constraint set (11) ensures with $b_{ej} = 0$ that an employee cannot perform a task if he/she does not possess the skills required. Moreover, the variable z_{jt} is added to the constraint to preclude work at a time when the task is not being processed. Constraint set (12) establishes the precedence relationships. Constraint set (13) defines over- and undercoverage for each task at each period. The auxiliary constraint sets (14) and (15) ensure that constraint set (13) is satisfied for every period. Constraint sets (16) and (17) limit the maximum possible under- and overcoverage, respectively. Constraint set (18) enforces that an employee can work at most in one task at each period. Constraint sets (19) and (20) are used to determine the CPT. Constraint set (21) forces z_{jt} to be 1 if and only if task j is active. Constraint set (22) ensures that task j must be performed ($z_{jt} = 1$) whenever it starts ($x_{jt} = 1$). Constraint

sets (23) and (24) model the nonpreemptive assumption: (23) determines that a task cannot be performed ($z_{jt} = 0$) after its completion time, and (24) states that it cannot be performed ($z_{jt} = 0$) before its starting time. Equations (25), (26), (27), and (28) define the variables employed in the model.

3.3.2. Extension 1: Limitation of Moves. The model presented above allows moving the staff from a task to another. These moves can be detrimental for the service company since they may involve setup and learning times as well as instability in the team composition that performs each task. This limitation of moves can be easily incorporated in the model adding the following two constraints:

$$y_{ejt} \geq w_{ej} - \left(1 - z_{jt}\right) \quad \forall e = 1, \ldots, E, \quad \forall j = 1, \ldots, J, \quad \forall t = 1, \ldots, T \tag{29}$$

$$y_{ejt} \leq w_{ej} \quad \forall e = 1, \ldots, E, \forall j = 1, \ldots, J, \tag{30}$$

where w_{ej} yields 1 if the employee e is assigned to the task j, 0 otherwise. Constraints (29) and (30) force that an employee have to be assigned to the task during the whole duration of the task when w_{ej} is equal to 1.

3.3.3. Extension 2: Generic Piecewise Linear Relationship. In Section 3.2 a piecewise linear relationship with two sections was presented to model the relationship between the processing time of a task and the amount of resources assigned to such task. In order to include a generic piecewise linear relationship with k sections, the constraints (31)–(38) must replace the constraints (19)-(20):

$$p_j = p_j^0 + \sum_{i=1}^{k} S_j^i \cdot us_j^i \quad \forall j = 1, \ldots, J \tag{31}$$

$$us_j^i \leq sa_j^i \cdot L^i + sl_j^i \cdot L^i \quad \forall i = 1, \ldots, k, \forall j = 1, \ldots, J \tag{32}$$

$$us_j^i > \left(sa_j^i - 1\right) \quad \forall i = 1, \ldots, k, \forall j = 1, \ldots, J \tag{33}$$

$$us_j^i < \left(2 - sa_j^i\right) \cdot L^i \quad \forall i = 1, \ldots, k, \forall j = 1, \ldots, J \tag{34}$$

$$us_j^i \geq sl_j^i \cdot L^i \quad \forall i = 1, \ldots, k, \forall j = 1, \ldots, J \tag{35}$$

$$\sum_{i=1}^{k} sa_j^i = 1 \quad \forall j = 1, \ldots, J \tag{36}$$

$$sa_j^i \leq sl_j^m \quad \forall j = 1, \ldots, J, \forall i, \frac{m}{m} < i \tag{37}$$

$$1 - sa_j^i \geq sl_j^m \quad \forall j = 1, \ldots, J, \forall i, \frac{m}{m} \geq i, \tag{38}$$

where sa_j^i is an auxiliary variable yielding 1 only in the section i, where us_j^i is over 0 and lower than L^i (represented by the constraints (33)-(34), (36)). Generic piecewise linear relationship is defined in constraint (31). sl_j^m is introduced to define with 1 the sections with $us_j^i = L^i$, 0 otherwise (constraint (35)). Finally, constraint (32) delimits the upper value of the variable us_j^i and the constraints (37)-(38) establish $sl_j^m = 1$ in each section m preceding the section, where $sa_j^i = 1$, $sl_j^m = 0$ otherwise. Thereby, any piecewise linear relationship could be considered in the model by means of these constraints. To adjust this relationship to the convex relationship with $c = 1$, it is necessary only define the parameter S^i replacing the term $p(u)$ by k/u in (2) (see the slope in (39) assuming that the length $L^i = L = (UL - LV)/k$ for each section i):

$$S^i = \frac{(k/(LV + L \cdot i)) - (k/(LV + L \cdot (i-1)))}{L}. \tag{39}$$

3.4. Additional Models

3.4.1. Model with Fixed Processing Times (FPT). In this section, the PSSA problem without CPT is analyzed. The goal is to compare the PSSA model with CPT (PSSA-CPT in the following) with the model with FPT (PSSA-FPT model). To the best of our knowledge, it is the first time that the PSSA problem is solved with processing times depending on the number of employees allocated, so it is relevant to assess the benefits of the assumption. Both models are similar, but there are two main changes. First, in the PSSA-FPT model, processing times do not depend on the number of employees assigned (i.e., this number is fixed and equal to the optimum). Second, over- and undercoverage are not allowed in this model; that is, h_j^+-, h_j^--, and ah_{jt}-related constraints are not taken into account. The constraint sets (13)–(17), (19)-(20) are then replaced by

$$\sum_{e=1}^{E} y_{ejt} = R_j \cdot z_{jt} \quad \forall j = 1, \ldots, J, \forall t = 1, \ldots, T. \tag{40}$$

Thus, the PSSA-FPT model consists of the constraint sets (6)–(12), (18), and (21)–(28), where the above p_j variable is now a data of the problem. Obviously, the PSSA-CPT problem is harder than the PSSA-FPT since it has more variables and constraints. It will be analyzed again in the computational results.

3.4.2. Formulation of the Model Using Discrete Time/Resource Trade-Off Problem (DTRTP) Approach with Staff Assignment. The model presented in this paper could be also modeled using a DTRTP approach adding the staff assignment and the release times constraints. However, the results found by this approach were considerably worse in computational time and in average relative percentage deviation. When both models were compared for each instance presented in Section 5.1 (several instances are shown in Table 3), only 363 optimal solutions were found using the DTRTP approach compared to the 567 of the proposed model and the computational time by the DTRTP approach presented a deviation of 287.1% over our proposed model (the deviation runtime is calculated considering only the instances which were optimal solutions found for both models; the amount of these instances is presented in the column of Table 3 denoted by "Both optimum") as well, justifying the PSSA-CPT model proposed here.

Table 3: Comparison of runtimes between models using DTRTP approach and PSSA with CPT approach.

Instances	DTRTP and staff assignment				PSSA with CPT				Both optimum	Deviation runtime
	#O	#F	#N	Runtime	#O	#F	#N	Runtime		
30-1.5-0.25-0.2	10	0	0	54.0	10	0	0	13.1	10	380%
30-1.8-0.50-0.5	10	0	0	330.2	9	0	1	177.2	9	266%
30-2.1-1.00-1.0	0	0	10	—	9	0	1	—	0	—
60-1.5-0.25-0.2	8	0	2	708	7	0	3	376.6	7	135%
60-1.8-0.50-0.5	1	0	9	—	6	0	4	—	0	—
60-2.1-1.00-1.0	0	0	10	—	5	1	4	—	0	—
Global	363	14	583	376.08	567	20	373	165.26	345	287.1%

Table 4: Some examples on the number of constraints and variables.

Instances	Constraints	Variables	Integer variables	Binary variables
30-1.5-0.25-0.2	93120	65701	60661	60480
30-1.8-0.50-0.5	156330	123031	117181	117000
30-2.1-1.00-1.0	237895	204241	198871	198690
60-1.5-0.25-0.2	433887	325801	305461	305100
60-1.8-0.50-0.5	633915	529561	510661	510300
60-2.1-1.00-1.0	1427967	1286641	1262821	1262460

3.5. Complexity of the Problem. The PSSA can be reduced to the identical parallel machine problem with two machines ($P2 \mid \text{prec}, 1 \leq p_j \leq 2, C_{\max}$ according to Graham et al. [46]), when $E = 2$, $b_{ij} = R_j = 1$, and $LV_j = UV_j = r_j = 0$ are considered. As this problem is NP-hard [47] and our problem is a generalization of it, the PSSA is NP-hard.

On the other hand, the constraint sets (6)–(23) approximately form a total of $9 \cdot J + 8 \cdot J \cdot T + E \cdot T + E \cdot J \cdot T + J \cdot J/2$ constraints. Regarding the variables, there are $6 \cdot J + 3 \cdot J \cdot T + E \cdot J \cdot T$ in the model, where $2 \cdot J \cdot T + E \cdot J \cdot T$ are binary (i.e., x_{jt}, z_{jt}, and y_{ejt}). The number of constraints and variables is calculated in Table 4 for some instances of the test beds presented in Section 5.1. Since the PSSA with CPT is an NP-hard problem and since the number of constraints and variables for medium-large size of the problem is very high, exact solvers for integer programming problems have difficulties to find feasible and optimal solutions in such instances and therefore approximate algorithms have to be implemented. In the next section, a simple GRASP algorithm is presented to get feasible solutions in lower computational times.

4. A Simple GRASP Heuristic Algorithm

Given the complexity of the problem, a greedy randomized adaptive search procedure (GRASP) is presented in this section to solve the PSSA problem with CPT. GRASP was first introduced by Feo and Resende [48] and it has been widely applied to several combinatorial optimization problems. A review of the application fields of this algorithm is done in Festa and Resende [49]. The algorithm consists of two phases: first a constructive phase is implemented, and then a local search phase is made to find better solutions in the neighborhood. The constructive phase is usually employed to find a feasible solution for the problem, while the local search phase is performed to improve the solution found in the constructive phase. Both phases are repeated until the stopping criterion is reached (see Procedure 1).

In our problem, fixed processing times are considered during the constructive phase. In the local search phase, the task schedule is fixed and more employees are gradually assigned to each task in order to decrease the processing times and thus possibly improving the makespan of the project. In summary, a project scheduling problem is solved in the constructive phase, a CPT problem in the local search phase, while the staff assignment problem is treated in both phases (see Figure 3). In the next subsections the phases are explained in detail.

4.1. Constructive Phase. The constructive phase consists in the successive allocation of employees and starting times to tasks. The goal is to assign the tasks as early as possible until all tasks are scheduled. In this phase, both the processing times and the number of assigned employees are kept fixed. The number of employees allocated to each task (ne_j) is randomly assigned within the upper and lower limits. The processing times of each task are calculated for the so-obtained number of employees using the expressions shown in Section 3. With respect to the task to be chosen at each step, a restricted candidate list (RCL) is defined based on the tasks' precedence relations. According to these relations, the tasks are classified into sets: set S_1 corresponds to tasks without predecessors, set S_2 is tasks which only have predecessors of set S_k, and, in general, set S_i contains tasks with predecessors in sets $S_{i-1}, S_{i-2}, \ldots, S_1$. The tasks are iteratively inserted in the schedule according to the following procedure. First, the RCL contains all tasks in set S_1. A task is randomly chosen from the RCL to be placed in the schedule. The process is repeated until the RCL is empty. Then, the tasks of set S_2 are moved to. The same procedure is carried out until each task is introduced in the schedule.

Once a task is chosen from the RCL, it has to be placed into the schedule. To do this, the earliest possible start time for the task is calculated. Next, the starting time of the task is fixed to this earliest time and employees (from those having the required skills for the task) are assigned until the

```
Procedure main()
    classify the tasks in sets (S_i) according to the precedence relations
    while the stopping criterion is not reached do
        constructionPhase()
        localSearchPhase()
        update the stopping criterion
    end
end
```

Procedure 1: Overall scheme of the GRASP algorithm.

```
Procedure constructivePhase()
    assign a random number of employees (ne_j) to each task j according to uniform
        distribution U[R_j − LV_j, R_j + LV_j]
    calculate the tasks processing times from the number of employees assigned
    TO:= tasksOrderRandom()
    EO:= employeesOrderRandom()
    buildSolution(ne_j, TO, EO)
end
```

Procedure 2: Constructive phase.

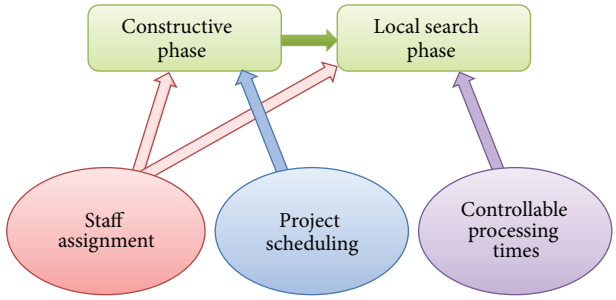


Figure 3: Division of the problem.

number of employees, ne_j, is reached. Employees are assigned one by one at random. At each step, the feasibility of the problem is checked. If, at some point, the resulting schedule is not feasible, the starting time of the task is postponed until feasibility is achieved. This process is repeated until the task is definitely inserted in a time period (see function *buildSolution*() in Procedure 3). It is important to note that the processing times are fixed at this phase. The steps of the constructive phase are shown in Procedure 2.

4.2. Local Search Phase. After obtaining a schedule of tasks and a staff assignment in the constructive phase, the local search phase aims to improve the solution by reducing the processing times of each task. To do so, tasks are first ordered again at random and then, step by step, the number of employees (ne_j) allocated to each task (following this order) is increased. In each step, the processing time of the task is updated and the makespan is analyzed (using the *buildSolution*() function, Procedure 3). If the makespan decreases, then the new solution is kept and the new amount of assigned employees is updated. Next, another employee is assigned to the same task and the objective function is analyzed. This process is repeated until the maximum overcoverage allowed for the task is reached. Then, we continue with the following task in the same way. The local search phase finishes when all tasks have been analyzed. The output of this phase is the best solution found. The pseudocode of this phase is shown in Procedure 4.

5. Computational Results

In this section, the problem under consideration is solved using the PSSA-CPT model introduced in Section 3 and the GRASP algorithm presented in Section 4. The PSSA-CPT model is implemented using the software Gurobi Optimizer 4.51 and ILOG CPLEX 12.4 and the GRASP heuristic is coded in C#. The computational experiments were tested on an Intel Core i7-930, 2.8 GHz, 16 GB RAM under Windows 7.

This section is divided into two parts. An explanation of the test bed is presented in Section 5.1 and, next, in Section 5.2, a comparison of the computational results discussed is made, along the following aspects:

(i) comparison between CPLEX and Gurobi solvers;

(ii) comparison between GRASP algorithm results and exact results;

(iii) impact of CPT on reducing the makespan of the project.

5.1. Test Bed. The integer programming problem presented in this paper was solved by the software Gurobi Optimizer 4.5 for instances based on the sets j30 and j60 of the PSPLIB classical instances [50], which are a reference test beds for project scheduling. Each combination of parameters is replicated ten times and the mean value is taken to represent it. In order to adapt the test beds to our problem, additional parameters have to be defined: maximum possible over- and

```
Procedure buildSolution(ne_j, TO, EO)
  foreach set S_i do
    foreach task according to TO do
      set the starting time of the task j (denoted x_j) according to the release time of
        this task and the completion times of the predecessors of the task, looking for
        the minimum possible starting time
      repeat
        assign employees, who are able to perform the task, to the task following
          the EO until the ne_j is reached
        if the assignment is unfeasible then
          x_j := x_j + 1
        end
      until the assignment is feasible;
    end
  end
  check the feasibility of the solution
end
```

Procedure 3: Function *buildSolution*().

```
Procedure localSearchPhase()
  set the best local solution as the solution from the constructive phase
  foreach task according to localSearchOrder do
    aux_ne_j := ne_j
    repeat
      aux_ne_j := aux_ne_j + 1
      buildSolution(aux_ne_j; TO; EO)
      if the solution is feasible and better than the best local solution then
        save this new solution as the best local solution
        ne_j := aux_ne_j
      end
    until the maximum overcoverage is reached;
  end
  check the feasibility of the solution
end
```

Procedure 4: Local search phase.

undercoverage (LV_j and UV_j) were drawn from uniform distributions $[0, R_j/2]$ and $[0, R_j]$, respectively; release times were chosen following uniform distributions $[0, T/6]$ and the number of employees was the average of the capacities of the 4 employees in the PSPLIB. The test bed used in this paper is similar to the test bed of Drezet and Billaut [17], where an employee was assigned a 70% probability to master a skill. In our paper, it is assigned an 85% probability to avoid infeasibility in the test beds. The parameters *kl* and *kr* are set to 2.5 according to the slope of the piecewise relationships explained in Section 3.2. In total, 960 instances were generated where 10 replicates were obtained for each combination of the parameters in Table 5 (see [51] for a detailed description of the parameters NC, RF, and RS).

Table 5: Test beds.

Parameters	Values
Number of tasks, j	30, 60
Network complexity, NC	1.5, 1.8, 2.1
Resource factor, RF	0.25, 0.50, 0.75, 1.00
Strength of the resource, RS	0.2, 0.5, 0.7, 1.0

5.2. Comparison and Analysis of Computational Results. The integer linear programming model was solved in this paper using the Gurobi software and ILOG CPLEX. Firstly, time limit was set to 30 min using Gurobi. The average results out of the 10 replicates for each parameter combination are shown in the 7th and 8th columns of Tables 6 and 7 (for the j30 and j60 instances, resp.). Dashed lines are used in some combination to specify that no solution was found for any replicate. The 9th, 10th, and 11th columns indicate the number of optimal, feasible, and nonfound solutions, respectively, for each combination of parameters found by Gurobi. It can be seen that optimal solutions were achieved for 567 instances from the total of 960 instances, feasible solutions were found for 16 cases, and no solutions before 30 minutes were therefore obtained for 377. The latter represents a huge amount of instances without solution and thus different method of resolution has to be implemented. More specifically, exact

Table 6: Experimental results for 30 tasks.

j-NC-RF-RS	CPLEX CPT					Gurobi CPT					Gurobi FPT			GRASP		
	C_{max}	t	#O	#F	#N	C_{max}	t	#O	#F	#N	C_{max}	t	dev.	C_{max}	ARPD	T
30-1.5-0.25-0.2	52.4	224.8	10	0	0	52.4	13.1	10	0	0	58.1	12.6	9.8%	52.4	0.00%	33.3
30-1.5-0.25-0.5	56.8	322.8	9	1	0	54.3	53.9	10	0	0	62.8	19.6	13.5%	54.4	0.20%	40.5
30-1.5-0.25-0.7	62.6	168.4	10	0	0	62.6	15.9	10	0	0	75.4	18.6	17.0%	62.6	0.00%	43.5
30-1.5-0.25-1.0	60.7	133.6	10	0	0	60.7	15.3	10	0	0	69.6	19.4	12.8%	60.7	0.00%	50.1
30-1.5-0.50-0.2	—	1800.0	0	6	4	—	1696.3	1	2	7	—	480.2	—	66.4	0.43%	40.5
30-1.5-0.50-0.5	—	1674.7	3	5	2	—	465.9	9	0	1	67.1	112.7	—	61.5	1.89%	57.7
30-1.5-0.50-0.7	—	1063.1	5	4	1	52.1	277.2	10	0	0	59.0	69.6	11.7%	52.5	0.76%	64.2
30-1.5-0.50-1.0	60.0	297.7	10	0	0	60.0	56.3	10	0	0	68.3	70.3	12.2%	60.0	0.00%	79.9
30-1.5-0.75-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1489.9	—	72.7	0.00%	44.4
30-1.5-0.75-0.5	—	1800.0	0	8	2	—	1366.1	4	0	6	—	322.1	—	61.6	1.21%	71.5
30-1.5-0.75-0.7	—	1664.6	2	4	4	—	696.8	8	0	2	72.0	104.1	—	64.2	1.95%	76.3
30-1.5-0.75-1.0	83.4	1163.5	6	4	0	—	459.9	9	0	1	64.5	92.2	—	57.2	0.22%	111.7
30-1.5-1.00-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	78.4	0.00%	51.4
30-1.5-1.00-0.5	—	1800.0	0	3	7	—	1800.0	0	0	10	64.1	359.6	—	61.9	0.00%	75.7
30-1.5-1.00-0.7	—	1653.9	1	4	5	—	1045.4	6	1	3	71.9	137.7	—	66.2	1.44%	90.4
30-1.5-1.00-1.0	—	1319.1	6	1	3	53.0	258.9	10	0	0	59.7	62.8	11.2%	53.0	0.00%	126.1
30-1.8-0.25-0.2	69.8	551.0	9	1	0	63.1	45.3	10	0	0	70.5	47.8	10.5%	63.2	0.17%	32.7
30-1.8-0.25-0.5	60.2	444.6	10	0	0	60.2	47.2	10	0	0	69.1	45.8	12.9%	60.4	0.39%	37.5
30-1.8-0.25-0.7	58.3	370.4	10	0	0	58.3	42.5	10	0	0	66.4	49.9	12.2%	58.3	0.00%	43.2
30-1.8-0.25-1.0	62.9	296.1	10	0	0	62.9	41.8	10	0	0	70.4	47.2	10.7%	62.9	0.00%	48.0
30-1.8-0.50-0.2	—	1800.0	0	6	4	—	1800.0	0	2	8	—	546.5	—	67.6	0.00%	37.5
30-1.8-0.50-0.5	—	1503.1	5	3	2	—	339.5	9	0	1	67.2	108.5	—	59.7	0.91%	55.3
30-1.8-0.50-0.7	—	1116.7	6	3	1	—	254.9	9	0	1	73.9	87.6	—	63.7	0.29%	58.9
30-1.8-0.50-1.0	—	978.2	7	2	1	58.7	114.2	10	0	0	68.2	90.8	13.9%	58.9	0.30%	73.0
30-1.8-0.75-0.2	—	1800.0	0	1	9	—	1800.0	0	0	10	—	1638.6	—	76.2	0.00%	42.6
30-1.8-0.75-0.5	—	1765.8	1	2	7	—	929.5	7	0	3	70.9	130.4	—	64.1	2.29%	65.5
30-1.8-0.75-0.7	—	1644.2	2	5	3	—	419.5	9	0	1	73.9	106.0	—	65.1	1.18%	75.1
30-1.8-0.75-1.0	79.6	973.8	8	2	0	65.2	150.5	10	0	0	74.5	115.6	12.5%	65.5	0.52%	89.2
30-1.8-1.00-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	96.4	0.00%	47.4
30-1.8-1.00-0.5	—	1800.0	0	3	7	—	1800.0	0	0	10	—	474.0	—	71.6	0.00%	74.2
30-1.8-1.00-0.7	—	1800.0	0	5	5	—	636.2	8	0	2	70.5	170.3	—	64.6	2.55%	87.1
30-1.8-1.00-1.0	—	1664.3	1	4	5	—	631.2	8	0	2	76.9	115.5	—	66.1	1.18%	107.2
30-2.1-0.25-0.2	63.0	823.0	9	1	0	62.2	232.9	10	0	0	71.0	67.5	12.4%	62.3	0.19%	31.5
30-2.1-0.25-0.5	64.0	680.6	9	1	0	64.0	59.7	10	0	0	72.7	60.3	12.0%	64.0	0.00%	36.9
30-2.1-0.25-0.7	66.1	546.3	10	0	0	66.1	47.0	10	0	0	75.6	61.1	12.6%	66.1	0.00%	41.4
30-2.1-0.25-1.0	61.7	455.6	10	0	0	61.7	41.0	10	0	0	73.4	66.2	15.9%	61.7	0.00%	47.1
30-2.1-0.50-0.2	—	1721.4	1	3	6	—	1470.4	2	0	8	—	434.5	—	71.3	0.77%	36.9
30-2.1-0.50-0.5	—	1613.5	3	5	2	—	667.3	8	0	2	77.1	94.5	—	67.7	1.74%	52.0
30-2.1-0.50-0.7	—	1506.5	3	3	4	63.1	367.4	9	1	0	72.3	103.3	12.7%	63.3	0.32%	58.3
30-2.1-0.50-1.0	—	1236.0	5	4	1	69.5	86.5	10	0	0	78.7	101.3	11.7%	69.7	0.30%	70.0
30-2.1-0.75-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1448.2	—	87.1	0.00%	40.9
30-2.1-0.75-0.5	—	1787.1	1	3	6	—	728.3	7	0	3	76.4	124.9	—	70.9	2.08%	60.4
30-2.1-0.75-0.7	—	1800.0	0	5	5	—	762.1	7	0	3	77.8	113.4	—	67.5	0.97%	67.3
30-2.1-0.75-1.0	—	1196.8	6	1	3	66.9	166.2	10	0	0	75.7	92.2	11.6%	66.9	0.00%	91.3
30-2.1-1.00-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	94.9	0.00%	44.5
30-2.1-1.00-0.5	—	1800.0	0	0	10	—	1800.0	0	1	9	—	1071.9	—	68.2	0.00%	69.1
30-2.1-1.00-0.7	—	1800.0	0	4	6	—	1096.2	6	0	4	75.4	156.1	—	69.6	1.08%	82.0
30-2.1-1.00-1.0	—	1470.8	5	3	2	—	539.7	9	0	1	69.7	111.9	—	61.7	0.28%	106.9
Average $j30$	—	1271.5	203	110	167	—	84.8	325	7	148	—	346.9	—	65.9	0.20%	61.8

Table 7: Experimental results for 60 tasks.

j-NC-RF-RS	CPLEX CPT					Gurobi CPT					Gurobi FPT			GRASP		
	C_{max}	t	#O	#F	#N	C_{max}	t	#O	#F	#N	C_{max}	t	dev.	C_{max}	ARPD	t
60-1.5-0.25-0.2	—	1800.0	0	1	9	—	803.6	7	0	3	109.9	246.5	—	100.3	0.30%	79.4
60-1.5-0.25-0.5	—	1800.0	0	3	7	96.7	434.0	10	0	0	107.4	198.2	10.0%	96.9	0.23%	112.4
60-1.5-0.25-0.7	—	1690.5	2	0	8	94.4	235.5	10	0	0	107.8	178.8	12.4%	94.9	0.54%	120.8
60-1.5-0.25-1.0	—	1719.7	2	2	6	99.4	101.5	10	0	0	112.8	167.1	11.9%	99.4	0.00%	153.9
60-1.5-0.50-0.2	—	1800.0	0	0	10	—	1670.2	1	0	9	—	1456.3	—	108.3	0.00%	96.8
60-1.5-0.50-0.5	—	1800.0	0	0	10	—	1193.5	7	1	2	109.2	431.5	—	100.6	0.88%	149.7
60-1.5-0.50-0.7	—	1800.0	0	0	10	—	461.2	9	0	1	109.1	220.4	—	97.0	0.10%	193.0
60-1.5-0.50-1.0	—	1800.0	0	0	10	98.5	285.1	10	0	0	108.8	226.2	9.5%	98.7	0.24%	245.3
60-1.5-0.75-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	128.9	0.00%	113.1
60-1.5-0.75-0.5	—	1800.0	0	0	10	—	1666.8	1	0	9	—	679.4	—	107.0	0.00%	189.9
60-1.5-0.75-0.7	—	1800.0	0	0	10	—	1514.0	5	1	4	—	491.7	—	93.6	0.65%	248.9
60-1.5-0.75-1.0	—	1800.0	0	0	10	146.0	1115.5	7	3	0	108.4	280.5	—	98.1	0.14%	355.2
60-1.5-1.00-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	139.1	0.00%	131.6
60-1.5-1.00-0.5	—	1800.0	0	0	10	—	1800.0	0	1	9	—	1259.9	—	99.3	0.00%	223.6
60-1.5-1.00-0.7	—	1800.0	0	0	10	—	1698.8	1	0	9	—	1220.5	—	101.5	0.33%	294.6
60-1.5-1.00-1.0	—	1800.0	0	0	10	—	1633.1	4	0	6	104.1	327.4	—	95.0	0.21%	398.6
60-1.8-0.25-0.2	—	1800.0	0	2	8	—	977.8	7	1	2	111.9	270.9	—	103.5	0.89%	73.9
60-1.8-0.25-0.5	—	1800.0	0	1	9	105.2	253.9	10	0	0	116.0	238.9	9.3%	105.2	0.00%	101.0
60-1.8-0.25-0.7	—	1765.7	1	3	6	103.5	132.0	10	0	0	113.9	186.3	9.1%	103.5	0.00%	113.0
60-1.8-0.25-1.0	—	1699.6	1	6	3	95.9	108.9	10	0	0	108.0	176.7	11.2%	95.9	0.00%	139.4
60-1.8-0.50-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1577.7	—	121.2	0.00%	90.8
60-1.8-0.50-0.5	—	1800.0	0	0	10	—	957.0	6	0	4	—	515.5	—	106.8	0.09%	149.8
60-1.8-0.50-0.7	—	1800.0	0	0	10	—	696.8	8	0	2	116.8	230.8	—	104.7	0.09%	171.4
60-1.8-0.50-1.0	—	1800.0	0	0	10	97.9	316.4	10	0	0	110.1	189.8	11.1%	98.1	0.19%	209.2
60-1.8-0.75-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	135.3	0.00%	105.8
60-1.8-0.75-0.5	—	1800.0	0	0	10	—	1737.1	1	1	8	—	1124.3	—	109.2	0.00%	176.7
60-1.8-0.75-0.7	—	1800.0	0	0	10	—	998.9	6	1	3	116.8	327.8	—	106.5	0.20%	211.0
60-1.8-0.75-1.0	—	1800.0	0	0	10	—	906.5	7	2	1	121.6	242.1	—	109.2	0.77%	282.6
60-1.8-1.00-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	159.7	0.00%	117.2
60-1.8-1.00-0.5	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1539.2	—	124.5	0.00%	207.5
60-1.8-1.00-0.7	—	1800.0	0	0	10	—	1729.2	1	0	9	—	942.2	—	109.7	0.00%	271.1
60-1.8-1.00-1.0	—	1800.0	0	0	10	—	1291.7	8	0	2	120.6	298.3	—	107.3	0.79%	321.6
60-2.1-0.25-0.2	—	1800.0	0	1	9	—	566.9	9	0	1	123.1	215.5	—	111.3	0.31%	73.3
60-2.1-0.25-0.5	—	1534.7	4	1	5	101.6	204.3	10	0	0	112.1	184.5	9.4%	101.7	0.09%	99.2
60-2.1-0.25-0.7	—	1575.9	2	2	6	104.5	151.3	10	0	0	117.1	175.9	10.8%	104.5	0.00%	110.0
60-2.1-0.25-1.0	—	1682.8	1	4	5	103.1	95.8	10	0	0	115.9	157.8	11.0%	103.1	0.00%	130.4
60-2.1-0.50-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1659.2	—	121.9	0.00%	87.2
60-2.1-0.50-0.5	—	1800.0	0	0	10	—	1417.8	4	0	6	—	558.0	—	105.8	0.85%	139.5
60-2.1-0.50-0.7	—	1800.0	0	0	10	—	621.6	8	0	2	116.2	221.2	—	106.5	0.52%	155.1
60-2.1-0.50-1.0	—	1800.0	0	0	10	108.3	233.5	10	0	0	122.0	192.4	11.2%	108.6	0.27%	195.4
60-2.1-0.75-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	146.3	0.00%	101.0
60-2.1-0.75-0.5	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1296.2	—	107.5	0.00%	159.9
60-2.1-0.75-0.7	—	1800.0	0	0	10	—	1741.5	2	0	8	124.7	355.5	—	111.5	0.17%	205.0
60-2.1-0.75-1.0	—	1800.0	0	0	10	—	718.1	8	0	2	114.0	208.5	—	101.9	0.40%	257.3
60-2.1-1.00-0.2	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1800.0	—	157.5	0.00%	113.6
60-2.1-1.00-0.5	—	1800.0	0	0	10	—	1800.0	0	0	10	—	1664.8	—	121.2	0.00%	188.8
60-2.1-1.00-0.7	—	1800.0	0	0	10	—	1800.0	0	1	9	—	1020.6	—	109.0	0.00%	239.2
60-2.1-1.00-1.0	—	1800.0	0	0	10	—	1304.8	5	1	4	120.3	240.8	—	108.3	0.54%	301.7
Average $j60$	—	1780.6	13	26	441	—	1112.0	242	13	229	—	712.4	—	110.1	0.53%	175.1
Global	—	1526.1	216	136	608	—	913.7	567	20	377	—	529.7	—	88.0	0.38%	118.5

solutions were achieved for 325 instances considering 30 tasks in the project, and only in 242 instances with 60 tasks. Using the same time limit than CPLEX, makespan and runtime are shown in the second and third columns. Optimal solutions were found only for 216 from the total of 960 instances (only 13 in the set j60) and feasible solutions for 352 instances and, hence, no solutions were found in 608 instances. This represents worse results as compared to that of Gurobi. With respect to the value of the makespan for the instances where feasible solutions were found, the ARPD (average relative percentage deviation) for CPLEX in these 352 instances was 58.4%, calculated using the expression (41), in contrast to the ARPD of 3.4% found by Gurobi in the 583 feasible solutions:

$$\mathrm{ARPD}_A = \frac{C_{\max}^{A} - \min\left(C_{\max}^{\mathrm{Gurobi}}, C_{\max}^{\mathrm{CPLEX}}, C_{\max}^{\mathrm{GRASP}}\right)}{\min\left(C_{\max}^{\mathrm{Gurobi}}, C_{\max}^{\mathrm{CPLEX}}, C_{\max}^{\mathrm{GRASP}}\right)}. \quad (41)$$

Finally, the GRASP heuristic was compared with the exact solutions. The time limit of the GRASP heuristic was set to $J \cdot E/10$ seconds; that is, it increases with the difficulty of the problem. Results are shown in Tables 6 and 7. Each row corresponds to a given combination of parameters and the first column indicates the chosen parameters in the format (j-NC-RF-RS). The column "$C_{\max}^{\mathrm{GRASP}}$" represents the average makespan using the GRASP algorithm for 10 iterations and the column "t^{GRASP}" is corresponding computation time ($J \cdot E/10$). The mean deviation between the solutions by Gurobi with time limit of 30 min and the solutions by GRASP with time limit of $J \cdot E/10$ are shown in the column "ARPD" using the expression (41).

The ARPD of the GRASP algorithm for all problems was 0.38%. Comparing the results of the exact solutions found by Gurobi (567 instances), the deviation of the solutions obtained by the GRASP algorithm is 0.62%, and optimal solutions using GRASP were found in 457 of these 567 instances (80.7% of the instances). Furthermore, feasible solutions were found by GRASP in each instance while feasible solutions were found in 587 instances using Gurobi.

Even better results are obtained when the GRASP algorithm is compared with Gurobi with the same time limit; that is, the time limit $J \cdot E/10$ is fixed for both GRASP heuristic and Gurobi. In this case, Gurobi found only optimal solutions in 201 of the total of 960 instances of the test bed while our heuristic found feasible solutions for all instances. Furthermore, the deviation of the heuristic from the optimal solutions was only 0.05% for these 201 instances and optimal solutions were found in 195 instances which represents a 97.01%.

In Section 3.4.1, the interest to compare the PSSA problem with CPT with the PSSA problem with FPT was discussed. The PSSA problem with FPT was solved by Gurobi setting also a time limit of 30 minutes. The makespan and runtime of this problem for each instance are shown in the 11th and 12th columns, respectively. Results of the PSSA-CPT and PSSA-FPT are compared in the 13th column. The objective function of PSSA-FPT represents an upper bound of the function objective of the PSSA-CPT ($C_{\max}^{\mathrm{PSSACPT}} \leq C_{\max}^{\mathrm{PSSA}}$), since a solution of PSSA-FPT is always a feasible solution for PSSA-CPT. The difference between the makespan and the runtimes of both models is analyzed. Considering only the instances with optimal solution (567 instances), the makespan decreases by 13.4% using CPT. On the other hand, runtime increased by 160.03%, which may seem an excessive increase in runtimes. However, since the planning horizon is always over 75 days, it seems reasonable that such a long and important decision must be performed carefully, so the runtime increase must not be so significant in that case.

6. Conclusions

In this paper, a project scheduling and staff assignment problem has been integrated. As in many real-life cases, the processing times of the tasks depend on the number of employees assigned; CPT were analyzed. There exists a feasible region where the relation of processing times and number of employees can be placed, so a piecewise linear relationship was proposed and included in a linear programming model for the problem in order to better represent the reality considering the penalty for communication in bigger teams and the penalty for lack of specialization in small teams. The consideration of a piecewise linear relationship between the processing times and the amount of resources in the PSSA-CPT provided a large reduction in runtime compared to the adaptation of the DTRTP adding staff assigned.

Different instances of the problem were first solved using the solver Gurobi Optimizer 4.5 and ILOG CPLEX 12.4, the former being more computationally efficient. Furthermore, a comparison between the model with CPT and with FPT was performed to justify the introduction of such CPT. The results obtained highlight an important decrease of the makespan by considering variable processing times. However, due to the complexity of the problem, extremely high computational times were needed for medium-large size instances by Gurobi, justifying the implementation of an approximate algorithm (GRASP). Computational times were greatly decreased there maintaining good ARPDs.

Regarding the future research lines of this paper, different approximate algorithms may be implemented in order to decrease the computational times comparing the results to the GRASP algorithm presented here and with other approximate algorithms employed for similar problems. Finally, a future research line relates to the CPT. In this paper, a piecewise linear relationship between processing times and number of employees (renewable discrete resources) has been introduced in the model. However, different relationships between both variables may be analyzed, such as a convex relationship and hyperbola, in order to adapt the model presented in this paper to continuous renewable and nonrenewable resources.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. J. Tersine and E. A. Hummingbird, "Lead-time reduction: the search for competitive advantage," *International Journal of Operations and Production Management*, vol. 15, no. 2, pp. 8–18, 1995.

[2] C. H. Glock, "Lead time reduction strategies in a single-vendorsingle-buyer integrated inventory model with lot size-dependent lead times and stochastic demand," *International Journal of Production Economics*, vol. 136, no. 1, pp. 37–44, 2012.

[3] E. Nowicki and S. Zdrzałka, "A survey of results for sequencing problems with controllable processing times," *Discrete Applied Mathematics*, vol. 26, no. 2-3, pp. 271–287, 1990.

[4] A. A. Fernandez, R. L. Armacost, and J. J. Pet-Edwards, "Understanding simulation solutions to resource constrained project scheduling problems with stochastic task durations," *Engineering Management Journal*, vol. 10, no. 4, pp. 5–13, 1998.

[5] B. L. Maccarthy and J. Liu, "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling," *International Journal of Production Research*, vol. 31, no. 1, pp. 59–79, 1993.

[6] M. H. A. Hendriks, B. Voeten, and L. Kroep, "Human resource allocation in a multi-project R&D environment: resource capacity allocation and project portfolio planning in practice," *International Journal of Project Management*, vol. 17, no. 3, pp. 181–188, 1999.

[7] A. Corominas, J. Ojeda, and R. Pastor, "Multi-objective allocation of multi-function workers with lower bounded capacity," *Journal of the Operational Research Society*, vol. 56, no. 6, pp. 738–743, 2005.

[8] A. Corominas, R. Pastor, and E. Rodríguez, "Rotational allocation of tasks to multifunctional workers in a service industry," *International Journal of Production Economics*, vol. 103, no. 1, pp. 3–9, 2006.

[9] S. P. Siferd and W. C. Benton, "Workforce staffing and scheduling: hospital nursing specific models," *European Journal of Operational Research*, vol. 60, no. 3, pp. 233–246, 1992.

[10] W. J. Abernathy, N. Baloff, J. C. Hershey, and S. Wandel, "A three stage manpower planning and scheduling model—a service sector example," *Abstracts of Hospital Management Studies*, vol. 9, no. 4, Article ID 09928, 30 pages, 1973.

[11] C. Tseng, C. Liao, and K. Huang, "Minimizing total tardiness on a single machine with controllable processing times," *Computers and Operations Research*, vol. 36, no. 6, pp. 1852–1858, 2009.

[12] D. Shabtay and G. Steiner, "A survey of scheduling with controllable processing times," *Discrete Applied Mathematics*, vol. 155, no. 13, pp. 1643–1666, 2007.

[13] E. L. Demeulemeester and W. Herroelen, *Project Scheduling*, Kluwer Academic Publishers, Boston, Mass, USA, 2002.

[14] O. Bellenguez and E. Néron, "Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills," in *Practice and Theory of Automated Timetabling V*, vol. 3616 of *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 229–243, Springer, 2005.

[15] O. Bellenguez-Morineau and E. Néron, "A branch-and-bound method for solving multi-skill project scheduling problem," *RAIRO—Operations Research*, vol. 41, no. 2, pp. 155–170, 2007.

[16] P. Brucker and S. Knust, *Complex Scheduling*, GOR-Publications, 2012.

[17] L.-E. Drezet and J.-C. Billaut, "A project scheduling problem with labour constraints and time-dependent activities requirements," *International Journal of Production Economics*, vol. 112, no. 1, pp. 217–225, 2008.

[18] G. L. Vairaktarakis, "The value of resource flexibility in the resource-constrained job assignment problem," *Management Science*, vol. 49, no. 6, pp. 718–732, 2003.

[19] V. Valls, Á. Pérez, and S. Quintanilla, "Skilled workforce scheduling in Service Centres," *European Journal of Operational Research*, vol. 193, no. 3, pp. 791–804, 2009.

[20] A. Corominas, J. Olivella, and R. Pastor, "A model for the assignment of a set of tasks when work performance depends on experience of all tasks involved," *International Journal of Production Economics*, vol. 126, no. 2, pp. 335–340, 2010.

[21] A. Drexl, "Scheduling of project networks by job assignment," *Management Science*, vol. 37, no. 12, pp. 1590–1602, 1991.

[22] B. Dodin and A. A. Elimam, "Audit scheduling with overlapping activities and sequence-dependent setup costs," *European Journal of Operational Research*, vol. 97, no. 1, pp. 22–33, 1997.

[23] C. Heimerl and R. Kolisch, "Scheduling and staffing multiple projects with a multi-skilled workforce," *OR Spectrum*, vol. 32, no. 2, pp. 343–368, 2010.

[24] R. Kolisch and C. Heimerl, "An efficient metaheuristic for integrated scheduling and staffing IT projects based on a generalized minimum cost flow network," *Naval Research Logistics*, vol. 59, no. 2, pp. 111–127, 2012.

[25] W. J. Gutjahr, S. Katzensteiner, P. Reiter, C. Stummer, and M. Denk, "Competence-driven project portfolio selection, scheduling and staff assignment," *Central European Journal of Operations Research*, vol. 16, no. 3, pp. 281–306, 2008.

[26] W. J. Gutjahr, S. Katzensteiner, P. Reiter, C. Stummer, and M. Denk, "Multi-objective decision analysis for competence-oriented project portfolio selection," *European Journal of Operational Research*, vol. 205, no. 3, pp. 670–679, 2010.

[27] M. Wu and S. Sun, "A project scheduling and staff assignment model considering learning effect," *International Journal of Advanced Manufacturing Technology*, vol. 28, no. 11-12, pp. 1190–1195, 2006.

[28] H. K. Alfares and J. E. Bailey, "Integrated project task and manpower scheduling," *IIE Transactions (Institute of Industrial Engineers)*, vol. 29, no. 9, pp. 711–717, 1997.

[29] M. Bassett, "Assigning projects to optimize the utilization of employees' time and expertise," *Computers and Chemical Engineering*, vol. 24, no. 2–7, pp. 1013–1021, 2000.

[30] R. M. Hachicha, E. M. Dafaoui, and A. El Mhamedi, "Assignment problem formulation under competence and preference constraints," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '10)*, pp. 1864–1868, December 2010.

[31] C. Heimerl and R. Kolisch, "Work assignment to and qualification of multi-skilled human resources under knowledge depreciation and company skill level targets," *International Journal of Production Research*, vol. 48, no. 13, pp. 3759–3781, 2010.

[32] J. Węglarz, J. Józefowska, M. Mika, and G. Waligóra, "Project scheduling with finite or infinite number of activity processing modes—a survey," *European Journal of Operational Research*, vol. 208, no. 3, pp. 177–205, 2011.

[33] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010.

[34] D. W. Pentico, "Assignment problems: a golden anniversary survey," *European Journal of Operational Research*, vol. 176, no. 2, pp. 774–793, 2007.

[35] J. N. D. Gupta, K. Krüger, V. Lauff, F. Werner, and Y. N. Sotskov, "Heuristics for hybrid flow shops with controllable processing

times and assignable due dates," *Computers and Operations Research*, vol. 29, no. 10, pp. 1417–1439, 2002.

[36] R. M. Belbin, *Management Teams*, Butterworth-Heinemann, Oxford, UK, 3rd edition, 2010.

[37] D. Shabtay and M. Kaspi, "Minimizing the total weighted flow time in a single machine with controllable processing times," *Computers and Operations Research*, vol. 31, no. 13, pp. 2279–2289, 2004.

[38] C. L. Monma, A. Schrijver, M. J. Todd, and V. K. Wei, "Convex resource allocation problems on directed acyclic graphs: duality, complexity," *Special Cases, and Extensions, Mathematics of Operations Research*, vol. 15, pp. 736–748, 1990.

[39] C. Lee and L. Lei, "Multiple-project scheduling with controllable project duration and hard resource constraint: some solvable cases," *Annals of Operations Research*, vol. 102, no. 1–4, pp. 287–307, 2001.

[40] M. J. Lanigan, "Task estimating: completion time versus team size," *Engineering Management Journal*, vol. 4, no. 5, pp. 212–218, 1994.

[41] P. C. Pendharkar and J. A. Rodger, "An empirical study of the impact of team size on software development effort," *Information Technology and Management*, vol. 8, no. 4, pp. 253–262, 2007.

[42] K. G. Smith, K. A. Smith, J. D. Olian, H. P. Sims Jr, D. P. O'Bannon, and J. A. Scully, "Top management team demography and process: the role of social integration and communication," *Administrative Science Quarterly*, vol. 39, pp. 412–438, 1994.

[43] M. A. Cusumano and R. W. Selby, *Microsoft Secrets*, The Free Press, New York, NY, USA, 1995.

[44] E. Demeulemeester, B. de Reyck, and W. Herroelen, "The discrete time/resource trade-off problem in project networks: a branch-and-bound approach," *IIE Transactions (Institute of Industrial Engineers)*, vol. 32, no. 11, pp. 1059–1069, 2000.

[45] M. R. Ranjbar and F. Kianfar, "Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms," *Applied Mathematics and Computation*, vol. 191, no. 2, pp. 451–456, 2007.

[46] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[47] J. K. Lenstra and A. H. G. R. Kan, "Complexity of scheduling under precedence constraints," *Operations Research*, vol. 26, no. 1, pp. 22–35, 1978.

[48] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[49] P. Festa and M. G. C. Resende, *GRASP: An Annotated Bibliography*, pp. 325–367, Kluwer Academic Publishers, Boston, Mass, USA, 2001.

[50] R. Kolisch and A. Sprecher, "PSPLIB—a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[51] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problems," in *Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems: Easy and Hard Instances*, Institute for Operations Research and the Management Sciences, 1992.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 949527, 10 pages
http://dx.doi.org/10.1155/2014/949527

Research Article

A Heuristic for Disassembly Planning in Remanufacturing System

Jinmo Sung and Bongju Jeong

Department of Information & Industrial Engineering, Yonsei University, 50 Yonsei-ro, Sinchon-dong, Seoul 120-749, Republic of Korea

Correspondence should be addressed to Bongju Jeong; bongju@yonsei.ac.kr

Received 24 January 2014; Accepted 11 March 2014; Published 23 April 2014

Academic Editors: E. K. Aydoğan, J. G. Barbosa, and D. Oron

Copyright © 2014 J. Sung and B. Jeong. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This study aims to improve the efficiency of disassembly planning in remanufacturing environment. Even though disassembly processes are considered as the reverse of the corresponding assembly processes, under some technological and management constraints the feasible and efficient disassembly planning can be achieved by only well-designed algorithms. In this paper, we propose a heuristic for disassembly planning with the existence of disassembled part/subassembly demands. A mathematical model is formulated for solving this problem to determine the sequence and quantity of disassembly operations to minimize the disassembly costs under sequence-dependent setup and capacity constraints. The disassembly costs consist of the setup cost, part inventory holding cost, disassembly processing cost, and purchasing cost that resulted from unsatisfied demand. A simple but efficient heuristic algorithm is proposed to improve the quality of solution and computational efficiency. The main idea of heuristic is to divide the planning horizon into the smaller planning windows and improve the computational efficiency without much loss of solution quality. Performances of the heuristic are investigated through the computational experiments.

1. Introduction

In a recent decade, most of environment-conscious industries have recognized the importance of remanufacturing where end-of-life products are collected and some useful parts are used again to remanufacture new parts. Remanufacturing may save resources on earth and production costs of company. As shown in Figure 1, end-of-life products are processed in remanufacturer to produce "as new" parts or disassembled to extract reusable parts which are furbished and used again by manufacturer.

In this paper, we deal with disassembly processes and especially focus on disassembly planning. Because disassembly is a preprocess of refurbishing or recycling, efficiency of it affects the whole remanufacturing system. In sustainable and environment-conscious industry, as the number of returned products increases, disassembly process becomes more significant and its related decision-making is getting complicated for most companies.

2. Related Literatures

As though disassembly planning problem in remanufacturing system looks like just the reverse of the conventional assembly planning problem, it is quite different in the purpose of retrieving necessary parts with the choice of efficient disassembly operations, which requires some different and unique approaches. Also, it deals with not only necessary parts but also unnecessary parts that are disposed or used later in future purpose. With all these reasons, disassembly planning problems is generally considered to be more complicated than assembly planning.

Previous researches on disassembly planning can be classified into three categories: full disassembly of a product with efficient sequence, disassembly to maximize the value of disposal parts, and disassembly to retrieve specific parts with minimum costs. The first category research aims to generate efficient or even optimal sequence of disassembly operations to disassemble a whole product. Most researches

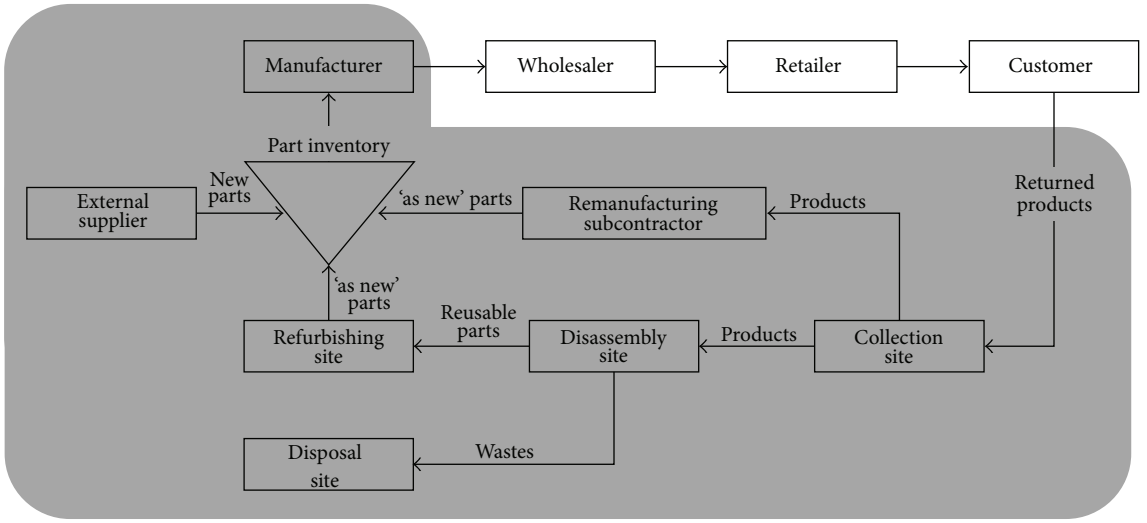


Figure 1: Conceptual framework of remanufacturing [22].

try to minimize disassembling costs because the costs are dependent on the selection of disassembly sequence. As one of the corner stone researches, Homem de Mello and Sanderson [1, 2] proposed a basic representation scheme of disassembly sequences with AND/OR graph, though it does not provide an optimal disassembly sequence. Lambert [3] proposed an approach to determine the optimal disassembly sequence using AND/OR graph scheme. He used Bellman's dynamic programming to select disassembly operations at each disassembly stage in backward achieving the minimum costs of whole disassembly costs. These researches have the limitation of application in remanufacturing system where only useful parts need to be obtained.

The second category of research focuses on maximizing the value of disassembled parts, which is very practical issue in remanufacturing system. The problem is to maximize total profit incurred by disassembly process by comparing the costs of disassembly to the value of all disassembled parts. Gupta and Taleb [4] emphasized that the purpose of disassembly process is not to manufacture as new products, but to meet the demand of useful parts, and they proposed a disassembly planning approach using MRP. Go et al. [5] reviewed many disassembly methods to enhance the recovery of end-of-life products, especially vehicle, so that valuable parts are efficiently retrieved. Lambert [6–8] proposed a mathematical model to find the optimal disassembly sequence with sequence-dependent setup costs using AND/OR graph representation. In order to solve this NP-hard problem, he employed iterative method which solves the relaxed problem without precedence constraints, checks the feasibility with added constraints, and then solves the problem with unsatisfied constraints in iterative manner. Although Lambert's method was proven to be very efficient in finding an optimal solution, it assumed only single planning period, single assembly product, and no demand for reusable parts. The last category is literally disassembly planning in remanufacturing environment, which means that it determines disassembly sequence and the amount of disassembly operations with minimum costs. Barba-Gutiérrez et al. [9] developed Reverse MRP using lot-sizing technique and analyzed the impact of lot-sizing technique on disassembly costs. Veerakamolmal and Gupta [10, 11] studied the effect of end-of-life disassembly by evaluating the design efficiency. They used a disassembly tree (DT) structure to represent the precedence relationships which show the order of disassembly parts. The major advantage of DT structure is to retrieve necessary parts at each branch point and make it possible to get demand of disassembly parts. However, as the number of parts in a product, the size of tree increases so rapidly. Furthermore, they [12] solved selective part recovery planning for electronic products using two-stage disassembly and retrieval system, even though the recovery planning is inapplicable to determine the schedule of disassembly operations. More recent researches consider disassembly planning over multiple periods [13–15], where the precedence among disassembly operations is constrained by part inventory and the schedule of disassembly operations is determined according to inventory costs and setup costs to meet the demand of parts. In these researches, however, since the disassembly operations are constrained by the amount of operations, not time constraints, the application to real environment may be very limited without adjustments.

For development of heuristic in this paper, we consider the rolling horizon planning technique which was firstly proposed by Modigliani and Hohn [16]; thereafter abundant applications followed. Chand et al. [17] emphasized the usage of the rolling horizon techniques including forecast and solution procedure in real operation problems based on time-series. They classified the previous operation planning researches by horizon type, model type, source of horizon, solution method, and subject and reviewed the techniques. Schwarz [18] used the rolling horizon planning technique to solve facility location problem, and other researchers such as Blackburn and Millen [19], Chand [20], and Federgruen and Tzur [21] applied it for inventory management problem. Most researches took advantage of rolling horizon technique in planning itself purpose. This study also employed the

rolling horizon technique, but for improving the efficiency of solution procedure instead of planning scheme.

The previous researches show that disassembly planning is more than just the reverse of assembly planning but should consider the recovery of useful parts/materials, impact of disposal on environment, and total costs as well. In this study, we are concerned about disassembly planning with efficiency and applicability in real remanufacturing environment. In order to do that, we make disassembly planning allow time-constrained operations, sequence-dependent setup, disassembled part demand, and multiple planning periods, and finally we propose an efficient heuristic to solve this NP-hard problem. This paper is organized as follows: We first develop a mathematical model of disassembly planning in Section 2 and propose a heuristic using rolling plan scheme in the following section. Computational experiments are performed using an example product to show the validity of our planning system and concluding remarks and discussion follow in Section 6.

3. Mathematical Model for Disassembly Planning

Disassembly planning in this study has the objective to minimize total disassembly costs of meeting part demands at each period. We recognize that this problem has the same property of the capacitated lot-sizing problem. Moreover, disassembly planning in remanufacturing has the demands of subassemblies at each period, not only the end parts in BOM (Bill-of-Material) structure. Also, since most workshops with disassembly process belong to small- or medium-size company which produces so many different types of products, we assume job-shop production environment rather than mass production line. This means that setup time and the related costs need to be included in the model. For the purpose of modeling, we assume the following production environment.

(i) The demands of parts/subassemblies are known at each period.

(ii) The unfulfilled demand through disassembly is fulfilled by purchase from vendor. The purchase cost is considered as penalty cost.

(iii) All setup and workload can be transformed to time units.

(iv) There exists a working available time at each production period.

(v) Setup can be carried over the next period, that is, the same operation can be carried over the next production period.

(vi) Transition matrix: T is a transition matrix with all disassembly operations (column) and subassemblies (row). The elements of T are interpreted as follows. For example in Table 1, disassembly operation 7 decomposes the subassembly BCD into CD and B with the values −1 (destruction) and 1 (creation) in the corresponding elements. Table 1 represents a transition matrix corresponding to a ball-pen of Figure 2.

With the above assumptions, the nomenclature shown at the end of the paper is needed for our model.

The objective function is represented as the sum of all relevant costs such as operation processing cost, inventory holding cost, setup cost, and purchasing cost. Then the model for disassembly planning is formulated as follows.

Model (P)

$$\begin{aligned}\text{Min} \sum_j \left(\text{pc}_j \times \sum_t X_{jt} \right) + \sum_i \left(\text{hc}_i \times \sum_t \text{Inv}_{it} \right) \\ + \sum_j \left(\text{sc}_j \times \sum_t Z_{jt} \right) + \sum_i \sum_t \text{pp}_i \times \text{Po}_{it}\end{aligned} \tag{1}$$

subject to

$$\begin{aligned}\text{Inv}_{it} = \text{Inv}_{i,t-1} - \sum_j \text{ps}_{ij} \times \left(\text{pr}_j \times X_{jt} \right) \\ + \sum_j \text{cs}_{ij} \times \left(\text{pr}_j \times X_{jt} \right) + \text{Po}_{it} - oq_{it} \quad \forall j, t\end{aligned} \tag{2}$$

$$p_j \times X_{jt} \le \sum_j \text{ps}_{ij} \times \text{Inv}_{i,t-1} \quad \forall j, t \tag{3}$$

$$\sum_j \text{st}_j \times Z_{jt} + \sum_j X_{jt} \le \text{timeCapa}_t \quad \forall t \tag{4}$$

$$\sum_j A_{jt} = 1 \quad \forall t \tag{5}$$

$$\sum_j B_{jt} = 1 \quad \forall t \tag{6}$$

$$X_{jt} \le M \times Z_{jt} \quad \forall j, t \tag{7}$$

$$A_{jt} \le Y_{jt} + W_t \quad \forall j, t \tag{8}$$

$$B_{jt} \le Y_{jt} + W_t \quad \forall j, t \tag{9}$$

$$A_{jt} - B_{j,t-1} \le Z_{jt} \quad \forall j, t \tag{10}$$

$$Y_{jt} - A_{jt} \le Z_{jt} \quad \forall j, t \tag{11}$$

$$B_{j,t-1} + W_t \le B_{jt} + 1 \quad \forall j, t \tag{12}$$

$$B_{j,t-1} + W_t \le A_{jt} + 1 \quad \forall j, t \tag{13}$$

$$\sum_j X_{jt} \le 1 - W_t \quad \forall t \tag{14}$$

$$\sum_j Y_{jt} \le 1 - W_t \quad \forall t \tag{15}$$

$$\sum_j Y_{jt} - 1 \ge (N - 1) \times \delta_t \quad \forall t \tag{16}$$

$$A_{jt} + B_{jt} \le 2 - \delta_t \quad \forall j, t \tag{17}$$

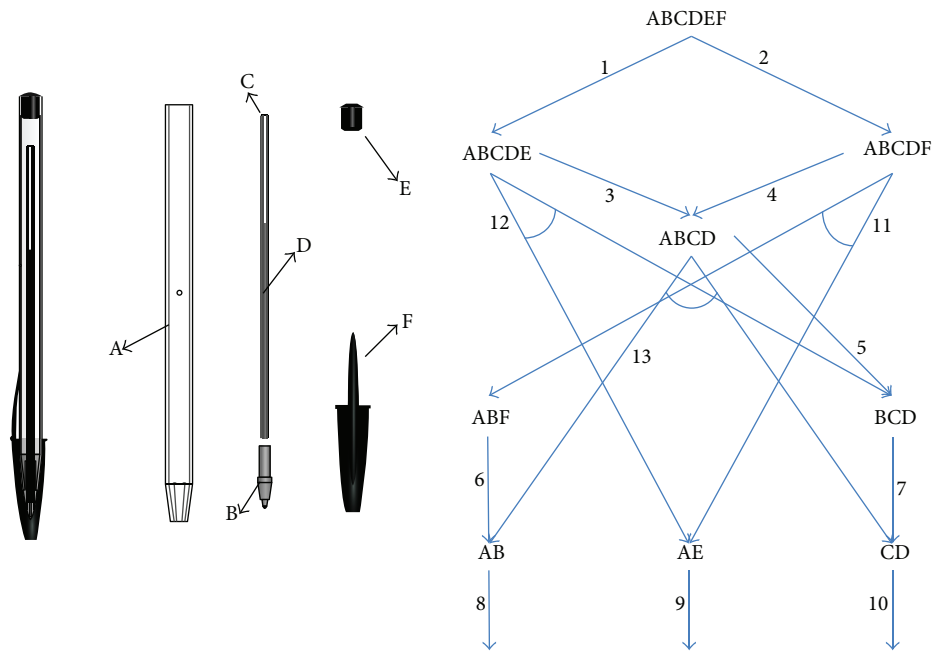


Figure 2: Disassembly AND/OR diagram of ball-pen [23].

Table 1: Disassembly transform matrix of ball-pen with 15 subassemblies.

	1	2	3	4	5	6	7	8	9	10	11	12	13
ABCDEF	−1	−1											
ABCDE	1		−1									−1	
ABCDF		1		−1							−1		
ABCD				1	−1								−1
ABF						−1					1		
BCD					1		−1					1	
AB						1		−1					1
AE									−1			1	
CD							1			−1	1		1
A								1	1				
B							1	1					
C										1			
D										1			
E		1	1						1				
F	1			1		1							

$$Y_{jt}, Z_{jt}, A_{jt}, B_{jt}, W_t \in \{0,1\} \quad \forall j,t \tag{18}$$

$$\mathrm{Po}_{it} \geq 0, \quad \mathrm{Inv}_{it} \geq 0 \quad \forall i,t \tag{19}$$

$$X_{jt} \geq 0 \quad \forall j,t \tag{20}$$

$$\delta_t \geq 0 \quad \forall t. \tag{21}$$

Constraints (2)–(4) ensure the sequence of subassembly operations to be performed. Constraint (2) is the balance equation of inventory between two consecutive periods. Constraint (3) ensures the precedence relationship among subassembly operations with regard to inventory of subassembly in each period. Constraint (4) imposes the total capacity limit in each period. Constraints (5)–(17) represent the requirements and conditions for operations and setups to be carried over the whole planning horizon. Constraints (5) and (6) indicate that each period should have the starting and finishing operations, respectively. Constraint (7) guarantees that disassembly operation without setup is not allowed over all periods. Constraints (8) and (9) ensure the feasible relationship among A, B, Y, and W variables. Constraints (10) and (11) indicate that over the planning horizon if the same operation is carried over the next period, there should be no setup. Constraints (12) and (13) represent that if there is no operation performed in the current period, the setup is already done in the previous period so that no setup is

necessary in the current period. Constraints (14) and (15) guarantee that the indicator variables of operations should have the valid values accordingly. Finally, constraint (16) and (17) ensure that if only one operation is performed in a period, it means that the first and last operation should be the same operation. The last set of constraints (18)–(21) represent the nonnegativity and binary conditions of all decision variables used in the model.

4. Heuristic

4.1. Basic Concept. In order to develop a heuristic for the problem presented in the previous section, we first examined the computation time for finding the optimal solution. Computational experiments were done using OPL Studio 6.0 by ILOG on the Pentium 4 2.8 Ghz Win-XP platform. We generated 21 types of problems by changing the size of demand (three types) and the length of planning horizon (seven types). Demand sizes are determined by the proportion of production capacity, that is, the order quantity of 10%, 20%, and 30% of production capacity, and the planning horizons are in the range of 4 to 10 planning periods, that is, total 6 types of planning horizon. For each type of problem, ten problems are generated using random number generator and total 210 problems were finally tested. Figure 3 shows the average computation time taken to solve each type of problem. The result shows no regular pattern according to the size of demand. However, increasing the length of planning horizon seems to make the computation time also increase. In particular, the computation time increases rapidly from planning horizon 7, while it stays very short until then.

The investigation indicates that the computation time for solving the mathematical model increases in highly nonlinear fashion as the length of planning horizon increases, but there seems to be no strong relationship with demand pattern. Based on the experiments, in order to overcome the problem with the long planning horizon in real production environment, we consider decomposing the original problem into the smaller size of subproblems with shorter planning horizon and then aggregating them back to get the final solution. Figure 4 illustrates an example of decomposition. For example, the original 10-periods problem is decomposed into four 7-periods subproblems where each subproblem shifts the planning horizon by one period until the last period is reached. Moreover, the modified cost parameters are adapted to make reasonable global solution. Thus, instead of solving one large problem, we solve four smaller problems separately with efficient time and integrate four solutions to make a solution for the original problem.

This procedure is known to be rolling horizon planning technique which was firstly proposed by Modigliani and Hohn [16]. However, since the subproblems are treated as an independent local problem, there should be a remedy to overcome the local optimality. In order to achieve global optimality for the original problem, we need to adjust setup and inventory holding costs for the local problem so that the local solution can be a part of global solution in the later aggregation procedure. This is because, basically, our model

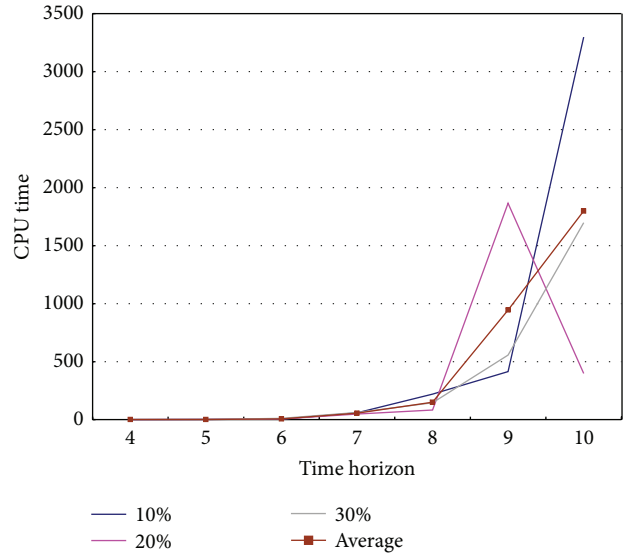


Figure 3: Computation time (sec) according to the length of planning horizon and demand quantity.

pursues the optimal trade-off between setup and holding costs. For example, if setup cost is much greater than holding cost in the original problem, we need to keep the number of setups small and the inventory as long as possible over the planning horizon. However, since the subproblem with short planning horizon cannot keep inventory for future demand, we adjust the setup cost to be smaller so that more operations are flexibly assigned for future demand. After solving all subproblems, the local solutions are aggregated to be global solution for the original problem.

4.2. Heuristic. Based on the concept described before, our heuristic consists of three steps: (1) adjusting setup cost, (2) solving subproblems, and (3) aggregation of solutions. Figure 5 presents the whole procedure of heuristic.

4.2.1. Adjusting Setup Cost. Setup costs for each subproblem are adjusted as follows.

(i) Set a fixed planning horizon of subproblem, such as $D < T$. In order to improve the quality of solution, D needs to be greater than the maximum number of operations for complete disassembly because the precedence relationship among subassembly operations is constrained by inventory of subassembly in each period as indicated in constraint (3) of our mathematical model.

(ii) Inventory holding cost of subassembly (hc_i) is transformed to operation-based cost (hc'_j) as follows:

$$\mathrm{hc}'_{(1\times j)} = \mathrm{hc}_{(1\times i)} \times \mathrm{rm}_{(i\times j)}. \tag{22}$$

(iii) New inventory cost (hc'_j) represents the changed holding cost by operation j. In other words, it is the

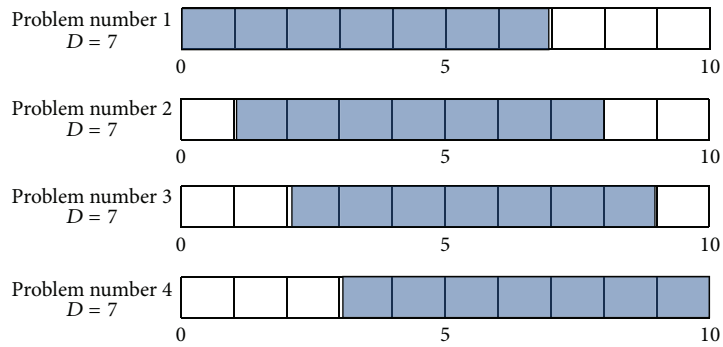


Figure 4: Generation of subproblems using rolling horizon technique.

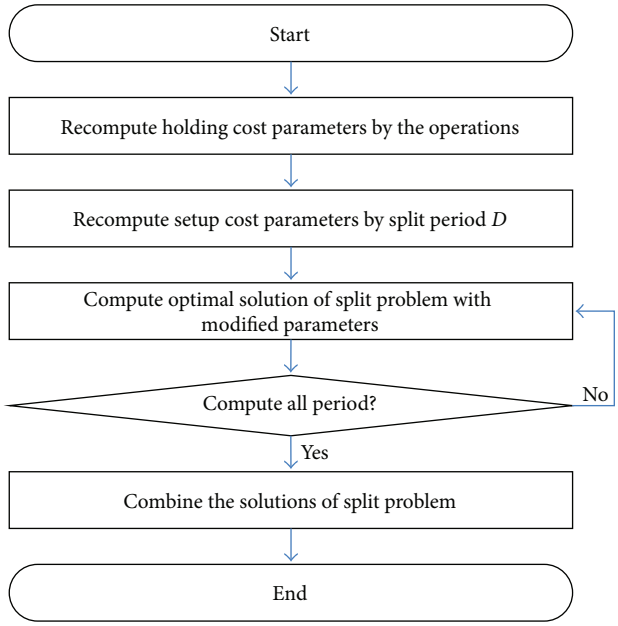


Figure 5: Procedure of heuristic.

difference between inventory holding cost before and after subassembly operation is performed.

(iv) Then, new setup cost (sc'_j) is computed as follows:

Case 1: If $sc/hc' < D$, $sc' = sc$,

Case 2: If $sc/hc' > D$, $sc' = hc' \times D$,

Case 3: If $hc' = 0$, $sc' = 1$.

4.2.2. Solving Subproblems. Using adjusted setup cost, each subproblem with planning horizon D is solved independently. For the feasibility of global solution, we need an additional constraint (23) as follows, which links all subproblems.

$$\text{Inv}_{i,0} = \text{initial\_Inv}_i \quad \forall i. \tag{23}$$

Then the model (P) is modified to the subproblem models (P') with the additional constraint (23) and adjusted setup costs. The solution procedure for model (P') using rolling horizon technique is presented as follows.

Set iteration, $n = 1$.

(i) The demand quantity for the nth subproblem is determined by the proportional quantity of total demand until period D.

(ii) Solve the problem $P'(n)$. If $P'(n)$ is the last subproblem, stop. Aggregate the solutions at the first period of each subproblem to make a global solution.

(iii) The inventory at the first period of $P'(n)$ becomes initial inventory of the next subproblem, $P'(n+1)$.

(iv) Set $n \leftarrow n+1$. Go to (i).

4.2.3. Aggregation of Solutions. Aggregation of subproblem solutions is simple. All the solutions are represented as the cost parameters of the original problem, but decision variables are replaced with the aggregated solution of subproblems as follows:

$$\sum_j \left(\text{pc}_j \times \sum_t R_{X\,jt} \right) + \sum_i \left(\text{hc}_i \times \sum_t R_{\text{Inv}\,it} \right) + \sum_j \left(\text{sc}_j \times \sum_t R\_Z_{jt} \right) + \sum_i \sum_t \text{pp}_i \times R\_\text{Po}_{it}. \tag{24}$$

The heuristic is programmed with pseudocode as shown in Pseudocode 1.

In the pseudocode, R_XX is solution for global solution and XX' is for subproblems.

5. Computational Experiments

In this section, the efficiency of our heuristic is investigated through computational experiments with the ball-pen example in Figure 2. Table 2 shows the data of unit holding cost of subassemblies and setup cost of operations. The various production environments are reflected by using cost structure and ease of disassembly. We consider two cost structures according to the relative percentage of processing cost out of setup cost as around 10% (Case A) and 40% (Case B). We assume that there are two types of products based on ease of disassembly, easy (product V) or difficult one (product W). Table 3 shows the unit processing cost of disassembly operations by Cases A and B, and Table 4 provides the unit

For $k = 1$ to $T - D - 1$ {
  For $d = 1$ to D {
    $oq'_{it} = oq_{i,k+d-1}$
    $initial\_Inv'_{i,0} = initial\_Inv_{i,0}$
  }
Solving (P')
$R\_X_{jk} = X'_{i,1}$
$R\_T_{jk} = T'_{j,1}$
$R\_Inv_{ik} = Inv'_{i,1}$
$R\_PO_{ik} = PO'_{i,1}$
$initial\_Inv_{i,0} = Inv_{i,1}$
}
$k = T - D$
$Ford = 1$ to D {
  $oq'_{it} = oq_{i,k+d-1}$
  $initial\_Inv'_{i,0} = initial\_Inv_{i,0}$
  }
Solving (P')
$R\_X_{j,t+k} = X'_{i,t}$
$R\_T_{j,t+k} = T'_{j,t}$
$R\_Inv_{i,t+k} = Inv'_{i,t}$
$R\_PO_{i,t+k} = PO'_{i,t}$
}

Pseudocode 1

Table 2: Unit holding cost (hc_i) and setup cost (sc_j).

(a)

Subassembly	ABCDEF	ABCDE	ABCDF	ABCD	ABF	BCD	AB	AE	CD	A	B	C	D	E	F
hc_i	0	9	8	6	6	8	5	6	5	4	4	2	1	1	1

(b)

Operation	1	2	3	4	5	6	7	8	9	10	11	12	13
sc_j	50	40	30	20	10	50	40	30	20	10	50	40	30

Table 3: Unit processing cost of operations (pc_j).

pc_j	1	2	3	4	5	6	7	8	9	10	11	12	13
A	5	4	3	2	1	5	4	3	2	1	5	4	3
B	20	16	12	8	4	20	16	12	8	4	20	16	12

purchase cost of subassemblies. In Table 5, production rate and setup time of operations are given.

All experiments were implemented using OPL Studio 6.0 by ILOG on the Pentium 4 2.8 Ghz Win-XP platform. For efficiency of experiments, we limited the run time at 5,400 seconds and over the limit the best value was taken. We considered three different planning horizons of 10, 15, and 20 periods. Demand data were generated according to the ordering probability of 10, 20, and 30%. For each ordering probability, two sets of data were generated. Therefore, 18 experiments (=6 types of demand $*$ 3 planning horizons) were performed for each cost structure (AV, AW, BV, and BW). The results are shown in Table 6 in terms of computation time and the quality of solution. The number of solution indicates the number of optimal solutions found by our heuristic out of 18 experiments. "Optimal" indicates the comparative results to the optimal solutions by OPL and "All" means the comparative performance to all solutions by OPL whether optimal or not.

From the results, we observe that the proposed heuristic finds a solution with very short computation time and the quality of solution is also quite good with less than 5% deviation from the optimal solution. We also find many cases where as the planning horizons increase the accuracy of solution is deteriorated. In order to improve it, one method is to make the subproblems with the longer planning horizon (D). Table 7 illustrates the results of this idea. For the 10-periods planning problem, solving the subproblems with 7-periods horizon provides the better cost results than 6-periods horizon subproblems. However, the increase of planning horizon results in the increase of computation time so rapidly. Thus determining the length of planning horizon for subproblems is the matter of trade-off between the accuracy and time efficiency.

Table 4: Unit purchase cost of subassemblies (pp_i).

pp_i	ABCDEF	ABCDE	ABCDF	ABCD	ABF	BCD	AB	AE	CD	A	B	C	D	E	F
A	0	58	56	52	46	44	42	40	38	34	32	30	28	26	24
B	0	174	168	156	138	132	126	120	114	102	96	90	84	78	72

Table 5: Production rate (pr_j) and setup time of operations (st_j).

Operation		1	2	3	4	5	6	7	8	9	10	11	12	13
V	pr_j	50	25	100	25	75	50	25	20	25	10	50	100	25
V	st_j	0.15	0.05	0.10	0.20	0.15	0.05	0.15	0.05	0.01	0.05	0.10	0.15	0.15
W	pr_j	40	25	25	40	60	70	45	40	100	90	55	70	65
W	st_j	0.20	0.15	0.10	0.20	0.10	0.20	0.05	0.20	0.15	0.15	0.20	0.05	0.05

Table 6: Performance results of heuristic.

Cost structure	Number of optimal solution	Avgerage computation time (sec)				Deviation from OPL solution			
		OPL		Heuristic		$D = 6$		$D = 7$	
		Average optimal	Average total	$D = 6$	$D = 7$	Optimal	All	Optimal	All
A, V	7	1183.9	Over 6983.1	57.8	156.6	0.96%	3.32%	0.74%	2.45%
A, W	6	675.1	Over 7223.9	107.5	252.8	3.00%	2.09%	3.04%	1.73%
B, V	7	916.5	Over 23246.9	95.2	376.7	2.30%	4.39%	1.44%	3.25%
B, W	10	20653	Over 27045	185.9	820.7	1.14%	1.97%	0.87%	1.43%

Table 7: Average objective function costs by OPL and heuristic.

(a)

	MIP				
	P	H	S	B	T
A, V	5658.4	6842.5	6645.8	4451.1	23597.8
A, W	5450.5	6214.0	7387.1	4702.6	23754.2
B, V	32158.1	6915.5	13451.7	6877.4	59402.7
B, W	34654.5	5677.2	13542.4	6150.3	60024.4

(b)

	$D = 6$				
	P	H	S	B	T
A, V	7726.2	4102.4	6699.3	5120.8	23648.7
A, W	9451.2	4412.3	5673.1	4672.4	24208.9
B, V	34248.6	5052.0	17536.1	4765.6	61602.4
B, W	35975.2	5735.0	14196.5	5471.9	61378.5

(c)

	$D = 7$				
	P	H	S	B	T
A, V	6314.5	6712.9	6513.1	4731.3	24271.8
A, W	5578.0	6325.3	7392.2	4825.8	24121.3
B, V	32574.5	6937.8	14782.9	6925.6	61220.9
B, W	35112.1	5752.3	13945.9	6184.2	60994.5

P: production cost, H: holding cost, S: setup cost, B: purchasing cost, and T: total cost.

In summary of experiments, our heuristic is quite good enough to be used in real disassembly planning system because of the high quality of solution and efficient computation time.

6. Conclusions and Discussion

Disassembly planning determines the sequence and schedule of disassembly operations to meet demand of disassembled subassemblies in remanufacturing environment. Unlike assembly planning, it is much more complicated due to reverse sequence generation and demand of in-process subassemblies. However, most previous researches have certain limits in applying to multiperiods disassembly planning with demand.

In this study, we formulated disassembly planning problem over multiperiods planning horizon with demands of subassemblies. Since the problem is known to be NP-hard, it requires tremendous computational time for reasonable size of problems in real production environments. For practical solution methods, we developed a heuristic based on rolling horizon planning technique which is useful in decomposing the large size problem and solving the subproblems efficiently. The heuristics has been investigated through computational experiments on a set of data reflecting various disassembly planning environments. The results show that our heuristic can be a good alternative method applicable to real production system in terms of computational efficiency and the quality of solution.

Remanufacturing system is still open area to be investigated and improved, especially for identifying many practical issues regarding sustainable environment. We need to continuously find those issues and provide good methods. Specifically, the concept behind the proposed heuristic in this

study can be applied to many planning problems found in remanufacturing system in order to improve the planning efficiency.

Index

i : Subassembly i
j: Disassembly operation j.

Parameter

T: The length of planning horizon, $t = 1, 2, \ldots, T$
oq_{it}: Order quantity of subassembly i in period t
pc_j: Processing cost of operation j per unit time
hc_i: Unit holding cost of subassembly i
sc_j: Setup cost of operation j
pp_i: Unit purchase cost of subassembly i
pr_j: Production rate of operation j (units/time)
rm_{ij}: –1 if subassembly i is disassembled by operation j, the number of subassemblies generated by operation j, otherwise
ps_{ij}: 1 if subassembly i is disassembled by operation j, 0 otherwise
cs_{ij}: The number of subassembly i produced by operation j ($\therefore rm_{ij} = cs_{ij} - ps_{ij}$)
st_j: Setup time for operation j
$timeCapa_t$: Available work time at period t.

Decision Variables

X_{jt}: Processing time of operation j during period t
Y_{jt}: 1 if operation j is performed during period t, 0 otherwise
Z_{jt}: 1 if there is a setup for operation j in period t, 0 otherwise
Inv_{it}: Inventory of subassembly i in period t
Po_{it}: The number of purchased subassembly i in period t
A_{jt}: 1 if operation j is ready for the first time in period t, 0 otherwise
B_{jt}: 1 if operation j is ready for the last time in period t, 0 otherwise
W_t: 1 if there is no operation to be performed in period t, 0 otherwise
δ_t: 1 if there is more than one operation to be performed in period t, 0 otherwise.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MEST) (no. 2009-0085893).

References

[1] L. S. Homem de Mello and A. C. Sanderson, "AND/OR graph representation of assembly plans," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 2, pp. 188–199, 1990.

[2] L. S. Homem de Mello and A. C. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 228–240, 1991.

[3] A. J. D. Lambert, "Optimal disassembly of complex products," *International Journal of Production Research*, vol. 35, no. 9, pp. 2509–2523, 1997.

[4] S. M. Gupta and K. N. Taleb, "Scheduling disassembly," *International Journal of Production Research*, vol. 32, no. 8, pp. 1857–1866, 1994.

[5] T. F. Go, D. A. Wahab, M. N. A. Rahman, R. Ramli, and C. H. Azhari, "Disassemblability of end-of-life vehicle: a critical review of evaluation methods," *Journal of Cleaner Production*, vol. 19, no. 13, pp. 1536–1546, 2011.

[6] A. J. D. Lambert, "Linear programming in disassembly/clustering sequence generation," *Computers and Industrial Engineering*, vol. 36, no. 4, pp. 723–738, 1999.

[7] F. J. D. Lambert, "Optimum disassembly sequence generation," in *Environmentally Concious Manufacturing*, vol. 4193 of *Proceedings of SPIE*, pp. 56–67, Society of Photographic Instrumentation Engineers, Boston, Mass, USA, February 2001.

[8] A. J. D. Lambert, "Optimizing disassembly processes subjected to sequence-dependent cost," *Computers and Operations Research*, vol. 34, no. 2, pp. 536–551, 2007.

[9] Y. Barba-Gutiérrez, B. Adenso-Díaz, and S. M. Gupta, "Lot sizing in reverse MRP for scheduling disassembly," *International Journal of Production Economics*, vol. 111, no. 2, pp. 741–751, 2008.

[10] P. Veerakamolmal and S. M. Gupta, "Design of an integrated component recovery system," in *Proceedings of the IEEE International Symposium on Electronics and the Environment (ISEE '98)*, pp. 264–269, May 1998.

[11] P. Veerakamolmal and S. M. Gupta, "High-mix/low-volume batch of electronic equipment disassembly," *Computers and Industrial Engineering*, vol. 35, no. 1-2, pp. 65–68, 1998.

[12] P. Veerakamolmal and S. M. Gupta, "Analysis of design efficiency for the disassembly of modular electronic products," *Journal of Electronics Manufacturing*, vol. 9, no. 1, pp. 79–95, 2000.

[13] D.-H. Lee and P. Xirouchakis, "A two-stage heuristic for disassembly scheduling with assembly product structure," *Journal of the Operational Research Society*, vol. 55, no. 3, pp. 287–297, 2004.

[14] H. Kim, D. Lee, and P. Xirouchakis, "A Lagrangean relaxation approach for capacitated disassembly scheduling," in *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA '05)*, pp. 722–732, May 2005.

[15] H.-J. Kim, D.-H. Lee, P. Xirouchakis, and O. K. Kwon, "A branch and bound algorithm for disassembly scheduling with assembly product structure," *Journal of the Operational Research Society*, vol. 60, no. 3, pp. 419–430, 2009.

[16] F. Modigliani and F. E. Hohn, "Production planning over time," *Econometrica*, vol. 23, no. 1, pp. 46–66, 1955.

[17] S. Chand, V. N. Hsu, and S. Sethi, "Forecast, solution, and rolling horizons in operations management problems: a classified bibliography," *Manufacturing and Service Operations Management*, vol. 4, no. 1, pp. 25–43, 2002.

[18] L. B. Schwarz, "A note on the near optimality of 5-EOQ's worth forecast horizons," *Operations Research*, vol. 25, pp. 533–536, 1977.

[19] J. Blackburn and R. Millen, "Heuristic lot-sizing performance in a rolling schedule environment," *Decision Sciences*, vol. 11, pp. 691–701, 1980.

[20] S. Chand, "A note on dynamic lot sizing in rolling horizon environments," *Decision Sciences*, vol. 13, pp. 113–119, 1982.

[21] A. Federgruen and M. Tzur, "Dynamic lot-sizing model with backlogging. A simple O(n log n) algorithm and minimal forecast horizon procedure," *Naval Research Logistics*, vol. 40, no. 4, pp. 459–478, 1993.

[22] K. Kim, I. Song, J. Kim, and B. Jeong, "Supply planning model for remanufacturing system in reverse logistics environment," *Computers and Industrial Engineering*, vol. 51, no. 2, pp. 279–287, 2006.

[23] A. J. D. Lambert, "Exact methods in optimum disassembly sequence search for problems subject to sequence dependent costs," *Omega*, vol. 34, no. 6, pp. 538–549, 2006.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 356364, 8 pages
http://dx.doi.org/10.1155/2014/356364

Research Article

Parallel-Batch Scheduling and Transportation Coordination with Waiting Time Constraint

Hua Gong, Daheng Chen, and Ke Xu

College of Science, Shenyang Ligong University, Shenyang 100159, China

Correspondence should be addressed to Daheng Chen; dahengchen@163.com

Received 23 January 2014; Accepted 20 February 2014; Published 15 April 2014

Academic Editors: J. G. Barbosa and D. Oron

Copyright © 2014 Hua Gong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper addresses a parallel-batch scheduling problem that incorporates transportation of raw materials or semifinished products before processing with waiting time constraint. The orders located at the different suppliers are transported by some vehicles to a manufacturing facility for further processing. One vehicle can load only one order in one shipment. Each order arriving at the facility must be processed in the limited waiting time. The orders are processed in batches on a parallel-batch machine, where a batch contains several orders and the processing time of the batch is the largest processing time of the orders in it. The goal is to find a schedule to minimize the sum of the total flow time and the production cost. We prove that the general problem is NP-hard in the strong sense. We also demonstrate that the problem with equal processing times on the machine is NP-hard. Furthermore, a dynamic programming algorithm in pseudopolynomial time is provided to prove its ordinarily NP-hardness. An optimal algorithm in polynomial time is presented to solve a special case with equal processing times and equal transportation times for each order.

1. Introduction

A supply chain is made of all the stages of value creation such as supply, production, and distribution. In logistics management, there are usually transportation of raw materials or semifinished products and distribution of products. Researches on supply chain management focus on developing strategies to help companies to improve optimal chain-wide performance by proper coordination of the different stages of supply chains. For an order, transportation of raw materials or semifinished products and production are two key operations in the supply chain system. A scheduling problem with the limited waiting time occurs when the processing of orders that finished transportation has to start in a given time. There are several industries where the transportation and production scheduling problem is influenced by the limited waiting time. Examples include fresh food and chemical and automobile industries. For instance, in case of fresh food production, canning operation must follow cooking operation to ensure freshness after transportation of raw materials. Additional applications can be found in the advanced automobile manufacturing environments such as just-in-time systems. Auto parts of orders are transported from suppliers to producers for further assembling. The producers must process the orders arrived at the factory in a given time in order to decrease the inventory level and increase the production level. This situation is relevant to logistics and supply chain management and specifically addresses how to balance the transportation rate of raw materials and the production rate.

Motivated by the aforementioned problem in logistics management, this paper describes a coordinated model for scheduling transportation of raw materials or semifinished products and parallel-batch production with waiting time consideration. There is a set of orders located at the different suppliers. The orders are transported by some vehicles to a manufacturing facility to be further processed. But only one order can be transported at a time. Each order arriving at the facility must be processed in its given waiting time. The orders are processed in batches on a parallel-batch machine, where a batch contains several orders and the processing time of the batch is the largest processing time of the orders in it. The goal is to find a schedule to minimize the sum of the total flow time and the production cost.

Production scheduling problems with transportation considerations have been studied by a number of researchers.

The parallel-batch machine scheduling problems have been previously studied in other manufacturing systems, especially burn-in operations in the very large-scale integrated circuit manufacturing (e.g., [1–3]). We briefly discuss some work related to integrated production scheduling and transportation decisions. Some research (see [4–7]) has been done for two machine flow shop problems featuring transportation of semifinished products and batching processing. We mention the studies in [4–7]; however, the problems there do not consider parallel-batch production or they consider another kind of batching with a constant processing time. The models in [8, 9] consider the coordinated scheduling problems with two-stage transportation and machine production that incorporate the scheduling of jobs and the pickup of the raw materials from the warehouse and the delivery arrangement of the finished jobs. The above scheduling problems do not consider the waiting time constraints. Another line of production scheduling models with transportation decisions focuses on the delivery of finished jobs to customers with a limited number of vehicles. Interested readers for the delivery coordination are referred to the recent review in [10]. The parallel-batch machine scheduling problems with batch delivery are considered in [11, 12]. All of the papers reviewed in this section deal in some way with the coordination of production scheduling and transportation at logistics and operations area, but none of them addresses and deals with the problem from the limited waiting time point of view. We not only consider the scheduling problem involving transportation capacity and transportation times but also take into account the order waiting time constraint and the batch capacity. Here, the coordination of batch processing, transportation, and waiting time constraints is a decision about whether or not to reduce production cost and improve machine utilization.

The remainder of this paper is organized as follows. In the next section, we introduce the notation to be used and describe the model. In Section 3, we analyze the optimal properties of the problem and prove strong NP-hardness of the general problem. In Section 4, a dynamic programming algorithm is provided to solve the problem with equal processing times and we prove that this case is NP-hard in the ordinary sense. Section 5 deals with a case with equal processing times and equal transportation times and shows how to find the optimal schedule. The last section contains a conclusion and some suggestions for future research.

2. Description and Notation

Our problem is formally stated as follows. Given a set of n orders (including some semifinished jobs or raw material in one order), $J = \{J_1, \ldots, J_n\}$, which are located at the different suppliers. The orders need to be transported to a manufacturing facility and processed on a single parallel-batch machine. Only m vehicles are available for transporting the orders, which can load only one order in one shipment. The vehicles are initially located at a transportation center. Let t_j denote the transportation time of order J_j from the supplier to the manufacturing facility. The transportation time from the factory back to the center is negligible. In the production part, each order J_j requires a processing time of p_i on the parallel-batch machine. The preemption of orders is not allowed. The parallel-batch machine can process a number of orders simultaneously as a batch as long as the total number of orders in the batch does not exceed the machine capacity c. The processing time of a batch is the longest processing time of all orders in it. Orders processed in the same batch have the same completion time, that is, their common start time (the start time of the batch in which they are contained) plus the processing time of the batch. Suppose that loading and unloading times are included in the transportation times and the processing times of orders. Associated with each order is w_j, which is the waiting time of an order for a period time from arriving at the machine to starting its processing on the machine. Here, w_j is restricted by a constant W where $\max_j p_j \le W \le \sum p_j$, such that the schedule has a feasible solution at least. Each batch requires a processing cost on the parallel-batch machine which reflects the production cost. The goal is to find a feasible schedule to minimize the sum of the total flow time and the total processing cost.

Decision Variables

Consider the following:

T_u: the total running time of vehicle u,

r_j: the arrival time of order J_j on the machine,

s_j: the starting time of order J_j on the machine,

w_j: the waiting time of order J_j on the machine, $w_j = s_j - r_j \le W$,

B_l: batch l,

b_l: the number of orders in B_l,

$P(B_l)$: the processing time of B_l on the machine, $P(B_l) = \max_{j \in B_l}\{p_j\}$,

S_l: the starting time of B_l on the machine,

D_l: the completion time of B_l on the machine,

C_j: the flow time of order J_j on the machine,

x: the number of batches,

$X(x)$: the total processing cost, a nondecreasing function of x,

$F(\pi) = \sum_{j=1}^{n} C_j + X(x)$: the objective function of a schedule π.

We follow the commonly used three-field notation, $\alpha|\beta|\gamma$, introduced to denote the problem under study. The problem is denoted by $Vm \rightarrow \text{batch}|w_j|\sum C_j + X(x)$ where α field indicates that the orders are first transported by m vehicles and then processed on the parallel-batch machine. β field describes the order restrictive requirement, and γ defines the objective function to be minimized.

Specifically, the tradeoff between processing and transportation gives rise to the sequence of transportation and the batch composition decisions. The schedule may produce more batches in order to satisfy the waiting time requirements of orders. This leads to the increase of the total processing

Table 1

Order	1	2	3	4	5	6	7	8
t_j	1	1	2	2	2	3	4	4
p_j	2	3	4	3	3	3	4	4

cost on the machine. On the other hand, if we create very few batches, then we may not obtain a feasible schedule due to waiting time constraint. Therefore, minimizing of the objective for the problem is achieved a feasible schedule for sequencing and batching to tradeoff the total flow time and the processing cost. The following example gives an illustration of this observation.

Example 1. Consider a set of 8 orders with their transportation times and processing times as shown below for the problem $Vm \rightarrow \text{batch}|w_j|\sum C_j + X(x)$.

Suppose $m = 2$, $c = 4$, $W = 4$, and $X(x) = 3x$ (see Table 1).

One vehicle transports the orders in turn: J_1, J_3, J_5, and J_7. Another vehicle transports the orders in turn: J_2, J_4, J_6, and J_8. The constructed schedule π consists of three batches, $B_1 = \{J_1, J_2, J_3, J_4\}$, $B_2 = \{J_5, J_6\}$, and $B_3 = \{J_7, J_8\}$, and has an objective value of 85. If another constructed schedule π' contains two batches $B_1 = \{J_1, J_2, J_3, J_4\}$, $B_2 = \{J_5, J_6, J_7, J_8\}$, then the waiting time of the order J_5 is 5. Although schedule π' has a smaller processing cost, it is not feasible. From these two schedules, we see that a minimized objective can be obtained from a proper combination of the sequence of transportation and the batching of processing under the waiting time constraints.

3. The Problem $Vm \rightarrow \text{batch}|w_j|\sum C_j + X(x)$

In this section, we analyze two properties of the scheduling problem and prove that $Vm \rightarrow \text{batch}|w_j|\sum C_j + X(x)$ is NP-hard in the strong sense by a reduction from the 3-Partition Problem, which is well known to be NP-hard in the strong sense [13].

3-Partition Problem (3-PP). Given $3h$ items, $H = \{1, 2, \ldots, 3h\}$, each item $i \in H$ has a positive integer size a_i satisfying $a/4 < a_i < a/2$ and $\sum_{i=1}^{3h} a_i = ha$, for some integer a. The question asked is whether there are h disjoint subsets $H_1, H_2, \ldots, H_h$ of H such that each subset contains exactly three items and its total size $\sum_{i \in H_j} a_i = a$, $j = 1, 2, \ldots, h$.

Lemma 2. *For the problem $Vm \rightarrow batch|w_j|\sum C_j + X(x)$, there exists an optimal schedule without idle time between orders on any vehicle.*

The proof of Lemma 2 is straightforward and is omitted.

The following result describes a candidate set of possible processing times.

Lemma 3. *For the problem $Vm \rightarrow batch|w_j|\sum C_j + X(x)$, there exists an optimal schedule in which each processing on the parallel-batch machine is made either at the arrival time of an order on the machine or immediately when the machine becomes available.*

Proof. Assume that there is a process which is scheduled neither at the arrival time of an order nor at a time when the machine becomes available. This process can be changed to the latest earlier time which fits either of those conditions. Since the same orders can be processed at that earlier time and there are no additional processes, the objective value is not increased. □

Theorem 4. *The problem $Vm \rightarrow batch|w_j|\sum C_j + X(x)$ is strongly NP-hard even if $m = 2$.*

Proof. To show the NP-hardness of the scheduling problem, we establish the following polynomial time reduction from the 3-PP. Given a 3-PP instance, we construct an instance for $Vm \rightarrow \text{batch}|w_j|\sum C_j + X(x)$ as follows.

Number of Orders. Consider

$$n = 7h + 7, \tag{1}$$

Ordinary Orders. Consider

$$t_j = \left\lceil \frac{j}{6} \right\rceil a_{\lfloor (1/2)(j+1) \rfloor}, \qquad p_j = 2\left(\left\lceil \frac{j}{6} \right\rceil + 1\right)a, \qquad j \in H = \{1, 2, \ldots, 6h\}. \tag{2}$$

Auxiliary Orders. Consider

$$t_j = 2(j - 6h)a, \quad j \in G = \{6h + 1, \ldots, 7h\}$$
$$p_j = p_{6h+i} = 2(i + 1)(a_{3i-2} + a_{3i-1} + a_{3i}), \quad j \in G = \{6h + 1, \ldots, 7h\}, \tag{3}$$
$$t_j = 0, \quad p_j = 2a, \quad j \in Y = \{7h + 1, \ldots, 7h + 7\},$$

Machine Capacity. Consider

$$c = 7, \tag{4}$$

Limited Waiting Time. Consider

$$W = 2ah, \tag{5}$$

Processing Cost Function. Consider

$$X(x) = \left[\frac{5}{3}a(h + 1)(h + 2)(h + 3)\right]^{x-h}, \tag{6}$$

Threshold Value. Consider

$$y = 4a(h + 1)(h + 2)(h + 3). \tag{7}$$

Clearly, in a solution to this instance of $Vm \rightarrow \text{batch}|w_j|\sum C_j + X(x)$ with the objective value not exceeding

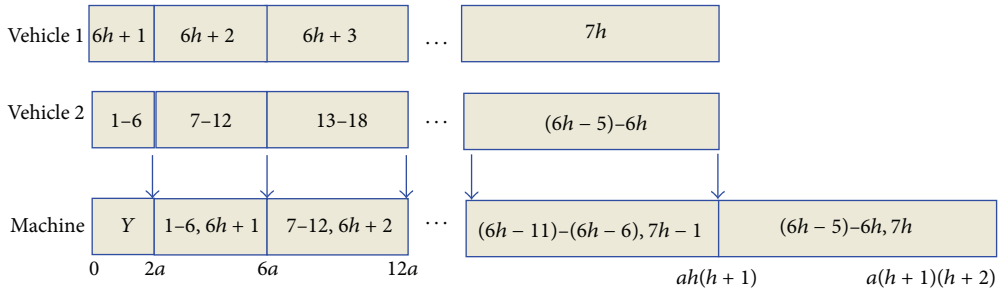


FIGURE 1: The schedule of Theorem 4.

y, we first prove the following properties: (1) since $t_j = 0$, for $j \in Y$, all the orders of Y as the first batch must be processed on the machine in the time interval $[0, 2a]$; (2) there will be exactly $h+1$ batches in the optimal schedule, and each batch contains 7 orders. Suppose that there are q (more than $h+1$) batches in the schedule. Then $7h+7$ orders need at least $h+1$ batches due to $c = 7$. If $q = h+2$, then the total processing cost is $X(x) = X(h+2) = [(5/3)a(h+1)(h+2)(h+3)]^2 > y$. Thus, all the orders of $H \cup G$ are divided into h batches; that is, all the batches are full.

We will show that the 3-PP instance has a solution if and only if there is a schedule π for the scheduling instance such that its objective value of π is no more than y.

$\rightarrow$ Assume that there is a solution to the 3-PP instance, $H_1, H_2, \ldots, H_h$, then there is a schedule to our problem with an objective value of no more than y. We construct a schedule π for our problem as shown in Figure 1.

In this schedule, two vehicles begin to transport orders at time point 0. All the orders of Y are processed first on the machine as the first batch. In the interval $[al(l-1), al(l+1)]$, vehicle 1 transports the orders of H_l, $H_l = \{6l-5, 6l-4, \ldots, 6l\}$, vehicle 2 transports J_{6h+l}, for $l = 1, 2, \ldots, h$. Since $t_{6h+l} = 2la$ and $\sum_{j \in H_l} t_j = 2la$, J_{6h+l} and the last order of H_l arrive at the machine simultaneously, for $l = 1, 2, \ldots, h$, then J_{6h+l} and the corresponding orders of H_l form batch B_l. The waiting time of each order in B_l is smaller than W. Hence, the starting time of B_l on the machine is $al(l+1)$; the completion time of B_l is $a(l+1)(l+2)$, for $l = 0, 1, 2, \ldots, h$. It is easy to see that the above schedule is optimal and the objective value is y.

$\leftarrow$On the contrary, suppose that there exists a schedule for the constructed instance with the objective value not exceeding y.

Fact 1. In the schedule, B_l contains an auxiliary order J_{6h+l}, for $l = 1, 2, \ldots, h$.

Assume that there is some batch B_l that contains more than one auxiliary order. Assume that B_l contains two auxiliary jobs J_{6h+l} and J_{6h+k}; then there must be some batch B_k only consisting of the ordinary jobs, $k \neq l$. Consider the following two cases.

Case 1. If $k = l-1$, then it is shown that $B_{l-1} = \{J_{6l-11}, \ldots, J_{6l-6}, J_{j'} \mid j' \in H_l\}$. The corresponding flow time of each order in B_{l-1} satisfies $C_j = al(l+1) + 2a$, for $j \in B_{l-1}$. The flow times of orders in each batch after B_{l-1} will be greater than $a(l+1)(l+2) + 2a$. It is easy to see that $\sum C_j = 4a(h+1)(h+2)(h+3) + 14(h-l+2)a$, $2 \leq l \leq h$. Hence, $F' = \sum C_j + X(h+1) > y$.

Case 2. If $k = l+1$, then it can be shown that two vehicles must transport two auxiliary orders of batch B_l, respectively. Due to the limited waiting time of orders, the completion time of batch B_l on the machine is greater than $a(l+1)(l+2) + 2a$. Furthermore, it is easy to see that $\sum C_j > 4a(h+1)(h+2)(h+3) + 14(h-l+1)a$, $1 \leq l \leq h-1$, which implies $F' > y$.

Fact 2. Each batch B_l must contain 6 orders of $H_l = \{6l-5, 6l-4, \ldots, 6l\}$ such that $\sum_{j \in H_l} t_j = 2la$, for $l = 1, 2, \ldots, h$.

By the above argument, since each batch is full, we can show that one vehicle transports the orders of H_l one by one, and another vehicle transports J_{6h+l} in the time interval $[al(l-1), al(l+1)]$. The orders $\{J_{6h+l}, H_l\}$ as B_l are processed on the machine in the time interval $[al(l+1), a(l+1)(l+2)]$. It must be true that $\sum_{j \in H_l} t_j = 2la$, for $l = 1, 2, \ldots, h$. Otherwise, if $\sum_{j \in H_l} t_j < 2la$, then $p_{6h+l} = 2(l+1) \cdot (a_{3l-2} + a_{3l-1} + a_{3l}) < 2(l+1)a$. We can obtain some batch B_k whose processing time is greater than $2(k+1)a$, $k \neq l$. Hence, $F' > y$, which is a contradiction.

Therefore, a partition for the set H is obtained by letting the elements be corresponding to batch B_l. Then it is easy to see that $q = h+1$ and h batches $B_1, \ldots, B_h$ form a solution to the 3-PP instance. The proof is concluded. □

Theorem 4 indicates that the existence of a polynomial time algorithm to solve the scheduling model is unlikely. Since the general problem is strongly NP-hard, we next consider two special cases of the problem.

4. The Problem $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$

We now turn our attention to a special case with identical processing time condition on the machine. In this section, we prove that the problem $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$ is ordinarily NP-hard by a reduction of the Partition Problem [13] which is a known NP-hard problem and provides a dynamic programming algorithm in pseudopolynomial time.

Suppose a feasible schedule $\pi = \{B_1, B_2, \ldots B_x\}$ with $x \in \{\lceil n/c \rceil, \ldots, n\}$ batches for the problem $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$. We now present some properties for the problem.

Lemma 5. *For $Vm \rightarrow batch|p_j = p, w_j|\sum C_j + X(x)$, there exists an optimal schedule π^* such that all orders assigned into the same vehicle are scheduled in the nondecreasing sequence of their transportation times.*

Proof. Assume that orders J_i and J_j are assigned to the same vehicle. J_i is followed by J_j immediately such that $t_i \geq t_j$ in π^*. Let π' be a schedule obtained by swapping J_i and J_j. In π', it is easy to see that $r_i > r'_j$ and $r'_i = r_j$. Regardless of whether J_i and J_j are processed in the same batch or not, the starting times of J_i and J_j on the machine cannot increase. We have $F(\pi^*) \geq F(\pi')$. □

Lemma 6. *For any scheduleof the problem $Vm \rightarrow batch|p_j = p, w_j|\sum C_j + X(x)$, the starting time of batch B_l satisfies $S_{l+1} \geq S_l + p$ for $l = 1, 2, \ldots, x$.*

Proof. It is easily obtained based on Lemma 3. □

Lemma 7. *For any schedule of the problem $Vm \rightarrow batch|p_j = p, w_j|\sum C_j + X(x)$, B_l contains all orders which have arrived at the machine if the number of the orders is no more than the capacity c in the time interval $(S_{l-1}, S_l]$.*

Proof. Assume that the batches are numbered in accordance with their start times, and the number of orders in each batch is smaller than the machine capacity c. Suppose that J_j is the first order assigned in B_{l+1} in π^*, and the arrival time of J_j on the machine satisfies $r_j \leq S_l$. Let π' be a new schedule obtained by simply assigning J_j to B_l. Then $C'_j = S_l + p < S_{l+1} + p \leq C_j$. The schedule of the remaining orders in π' are the same as in π^*. It is obvious that $F(\pi') \leq F(\pi^*)$. We can see that there exists an optimal schedule in which all batches consist of a number of orders which finish processing contiguously. □

We will show that the problem with equal processing times is NP-hard by a reduction from the Partition Problem.

Theorem 8. *The problem $Vm \rightarrow batch|p_j = p, w_j|\sum C_j + X(x)$ is NP-hard even if $m = 2$.*

Proof. We prove this result by reducing the Partition Problem to $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$. The Partition Problem (PP) can be stated as follows: given h items, $H = \{1, 2, \ldots, h\}$, each item $i \in H$ has a positive integer size a_i, such that $\sum_{i=1}^{h} a_i = 2a$, for some integer a. The question asked is whether there are two disjoint subsets H_1 and H_2, such that $\sum_{i \in H_1} a_i = \sum_{i \in H_2} a_i = a$.

We construct a corresponding instance of the problem $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$ as follows.

Number of Orders. Consider

$$n = 2h. \tag{8}$$

Transportation Times. Consider

$$\begin{aligned} t_j &= a_j, \quad j \in H = \{1, 2, \ldots, h\}; \\ t_j &= 0, \quad j \in Y = \{h+1, h+2, \ldots, 2h\}. \end{aligned} \tag{9}$$

Processing Times. Consider

$$p_j = a, \quad j = 1, 2, \ldots, 2h. \tag{10}$$

Limited Waiting Time. Conider

$$W = a. \tag{11}$$

Processing Cost Function. Consider

$$X(x) = 3ahx. \tag{12}$$

Machine Capacity. Consider

$$c = h. \tag{13}$$

Threshold Value. Consider

$$y = 9ah. \tag{14}$$

First, it is easy to see that, in a solution to this instance of $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$ with the objective value not exceeding y. We first prove the following property: since $t_{h+1} = \cdots = t_{2h} = 0$, all the orders of Y as the first batch must be processed on the machine in the time interval $[0, a]$. Now, we prove that there is a solution to the constructed instance of $Vm \rightarrow \text{batch}|p_j = p, w_j|\sum C_j + X(x)$ with total cost not exceeding y if and only if there is a solution to the PP instance.

$\rightarrow$ If there is a solution to the PP instance, we show that there is a schedule for the above-constructed instance with an objective value of no more than y. Let H_1 and H_2 be subsets of H that solves the PP instance. Vehicle 1 and Vehicle 2 transport the orders of H_1 and the orders of H_2, respectively. Let T_u be the total running time of vehicle u, for $u = 1, 2$. Since $\sum_{j \in H_1} t_j = \sum_{j \in H_2} t_j = a$, we have $T_1 = T_2 = a$. In fact, the completion time of the first batch on the machine is a. We can obtain that the waiting time of each order on the machine is not greater than a. Let $B_1 = Y$ and $B_2 = H$. It is easy to see that $b_1 = b_2 = h$, and the total flow time of all the orders is $3ha$. Hence, the objective is y.

$\leftarrow$Given a schedule with an objective value not exceeding y, then there must be a solution to the PP instance. We can obtain that all the orders in H are processed in the second batch on the machine; that is, all the orders are divided into two batches and $b_1 = b_2 = h$. Suppose that the number of the batches is greater than 2. Note that even if the waiting time constraint is ignored, the objective value is $F \geq ah + \sum_{j \in H} C_j + 9ah > y$, which is a contradiction. Thus, $\pi = \{B_1, B_2\}$ and $B_1 = Y$ and $B_2 = H$.

If $T_1 = \sum_{j\in H_1} t_j < a$, then it implies $T_2 = \sum_{j\in H_2} t_j > a$. Hence, the starting time of batch B_2 is $S_2 = T_2$. It implies that the total flow time is $\sum C_j = ah + (S_2 + a)h > 3ah$. We have $F > y$, which is a contradiction. Thus, $\sum_{j\in H_1} a_j = \sum_{j\in H_2} a_j = a$. □

Let $T = \sum_{j=1}^{n} t_j$. In the following, we derive a dynamic programming algorithm in pseudopolynomial time to solve the problem $Vm \to \text{batch}|p_j = p, w_j|\sum C_j + X(x)$.

Algorithm D 1. Index the orders in the nondecreasing transportation time sequence; that is, $t_1 \le t_2 \le \cdots \le t_n$.

Define $H_R(j, i, l, l, T_1, T_2, \ldots, T_m)$ and $f_R(j, i, l, l, T_1, T_2, \ldots, T_m)$ as the minimum objective value and the minimum total flow time of a partial schedule of orders $J_1, \ldots, J_j$, respectively, where orders $J_1, \ldots, J_j$ have finished transporting and processing by using l batches on the machine, and the current batch B_l contains orders $J_i, \ldots, J_j$.

Initial Conditions. Consider

$$H_x(j, i, l, T_1, T_2, \ldots, T_m) = \begin{cases} 0, & j = 0, i = 0, l = 0, T_1 = T_2 = \cdots T_m = 0 \\ \infty, & \text{otherwise,} \end{cases} \quad (15)$$

for $S_1 \ge t_1$, $l = 1, 2, \ldots, x$, $x \in \{\lceil n/c\rceil, \ldots, n\}$, and $T_u = 0, 1, \ldots, T$.

Recursive Relations. Consider

$$f_x(j, i, l, T_1, T_2, \ldots, T_m) = \begin{cases} \min_{1\le u\le m} \left\{f_x\left(j-1, i, l, T_1, \ldots, T_u - t_j, \ldots, T_m\right) + C_j\right\} & \text{if } b_l < c \\ \infty & \text{if } b_l \ge c, \end{cases} \quad (16)$$

where $C_j = \{S_l + p | S_{l-1} < T_u \le S_l \le T_u' + W, l = 1, 2, \ldots, x\}$, for $S_l + p \le S_{l+1} \le T_u' + W$, $l = 1, 2, \ldots, x$, $x \in \{\lceil n/c\rceil, \ldots, n\}$, and $T_u' = \min_{S_{l-1}<T_u\le S_l}\{T_u\}$.

Optimal Solution. Consider

$$F^* = \min\left\{H_x(n, n+1, l, T_1, T_2, \ldots, T_m) + X(x)\right\}. \quad (17)$$

Theorem 9. *Algorithm D1 can find an optimal schedule for the problem $Vm \to batch|p_j = p, w_j|\sum C_j + X(x)$ in $O(mn^3T^{m-1}W)$ time.*

Proof. Based on Lemma 5, there exists an optimal schedule with orders assigned to each vehicle in the nondecreasing transportation time sequence. If order J_j is assigned to vehicle u, then its starting time on the machine satisfies $r_j + W \le T_u$. If the number of orders in the current batch B_l is less than c, order J_j can be assigned into B_l. The corresponding flow time increases $S_l + p$. Based on Lemma 3, we have $S_{l-1} < T_u \le S_l \le T_u' + W$. This shows that Algorithm D1 can find optimal solution for the problem.

The time complexity of the algorithm can be established as follows. We observe that $i, j, l \le n$, $T_u \le T$, and $m - 1$ of the value $T_1, T_2, \ldots, T_m$ are independent. Thus, the number of different states of the recursive relations is at most n^3T^{m-1}. For each state, each use requires $O(mW)$. Therefore, the overall time complexity of Algorithm D1 is $O(mn^3T^{m-1}W)$. □

The existence of such a pseudopolynomial time algorithm for a NP-hard problem means that the problem is NP-hard in the ordinary sense. We have the following theorem.

Theorem 10. *The problem $Vm \to batch|p_j = p, w_j|\sum C_j + X(x)$ is ordinarily NP-hard.*

5. The Problem $Vm \to \text{batch}|p_j = p, t_j = t, w_j|\sum C_j + X(x)$

In this section, we consider a special case with identical processing times and identical transportation times and derive a polynomial-time algorithm to solve it. It is evident that the problem reduces to an optimal batching problem on the machine for the special case. Note that x is a decision variable denoting the number of batches on the machine. Note that $n_0 = \lceil n/m\rceil$ and $g = n_0m - n$; let n_u be the number of orders transported on vehicle u under a specific schedule. Note that $n_0' = \lceil n/c\rceil$ and $n_0' \le x \le n_0$, if $c \ge m$. Hence, the number of batches x on the machine can be numbered by n_0', $n_0' + 1, \ldots, n_0$. Otherwise, $n_0 \le x \le n_0'$ if $c < m$. Without loss of generality, assume that $c \ge m$ in the following discussion. Let $\pi^* = (B_1, B_2, \ldots B_L)$ be an optimal schedule with given $x = L$ batches for the problem $Vm \to \text{batch}|p_j = p, t_j = t, w_j|\sum C_j + X(x)$. Then we have the following lemma.

Lemma 11. *For the problem $Vm \to batch|p_j = p, t_j = t, w_j|\sum C_j + X(x)$, there exists an schedule with L batches in which* (1) $n_0 - 1 \le n_u \le n_0$, $u = 1, 2, \ldots, m$, (2) $\bar{b}_l = b_l/m$, $\bar{b}_l$ *is integral, and* $l = 1, 2, \ldots, L-1$, *and* (3) $S_{l+1} = \max\{r_j, S_l + p\}$, *where r_j denotes the arrival time of some m jobs on the machine,* $l = 1, 2, \ldots, L-1$.

Proof. (1) Based on Lemma 2, each vehicle has no inserted idle time during transporting. Assume that there exists an optimal schedule π^* in which the condition is not satisfied. Then there must be a pair of vehicles u and v such that $n_u \ge n_v + 2$ and the last order on vehicle u is processed in the last batch. If the last order on vehicle u is moved to the last position on vehicle v, then the objective value will not increase. By repeating this process, we can obtain a desired optimal schedule.

(2) According to Lemma 7, in the optimal schedule, B_l contains all orders which finish transportation in the time interval $(S_{l-1}, S_l]$, if the number of orders in a batch is no more than c. Since $t_j = t$ for each order J_j, m orders arrive at the machine together. A batch contains m orders or km ($km \le c$) orders, if their waiting times are not greater than the limited value W. Hence, $\bar{b}_l$ is integral, for $l = 1, 2, \ldots, L-1$.

(3) It is trivial according to Lemma 3. □

Based on these results, we can easily construct an optimal schedule for the problem.

Algorithm D 2. Calculate $n_0 = \lceil n/m \rceil$, $n_0' = \lceil n/c \rceil$.

For each $L = n_0', n_0' + 1, \ldots, n_0$, compute $\overline{b}_l = \overline{b}_0 - 1$, $l = 1, \cdots, h$, $\overline{b}_l = \overline{b}_0$, and $l = h + 1, \ldots, L$, where $\overline{b}_0 = \lceil n_0/L \rceil$ and $h = \overline{b}_0 L - n_0$.

Consider the following two cases.

Case 1. Consider $h = 0$.

If $(\overline{b}_0 - 1)t > W$ or $m\overline{b}_0 > c$, then the constructed schedule is not feasible, and denote $F(\pi_L^*) = \infty$. If $(\overline{b}_0 - 1)t \le W$ and $m\overline{b}_0 \le c$, then $S_1 = \overline{b}_0 t$. According to Lemma 6, there must be $\overline{b}_0 tl + t + W \ge S_l + p$. Otherwise, it is not feasible. Based on Lemma 7, we can obtain

$$S_{l+1} = \max\left\{\overline{b}_0 t\,(l+1), S_l + p\right\}, \quad l = 1, 2, \ldots, L-1. \tag{18}$$

The associated objective value can be derived as

$$F_1(\pi_L^*) = \min\left\{\sum_{l=1}^{L} m\overline{b}_0\,(S_l + p)\right\} - g\,(S_L + p) + X(L). \tag{19}$$

Case 2. Consider $h > 0$.

If $(\overline{b}_0 - 2)t > W$ or $m\overline{b}_0 > c$, then it is not feasible, and denote $F(\pi_L^*) = \infty$. If $(\overline{b}_0 - 2)t \le W$ and $m\overline{b}_0 \le c$, then we can show $S_1 = (\overline{b}_0 - 1)t$. According to Lemma 6, we have

$$\begin{aligned} &\left(\overline{b}_0 - 1\right) tl + t + W \ge S_l + p \quad \text{if } l \le h \\ &\left(\overline{b}_0 - 1\right) th + \overline{b}_0 t\,(l - h) + t + W \ge S_l + p \quad \text{if } l > h. \end{aligned} \tag{20}$$

Otherwise, the constructed schedule is not feasible, and denote $F(\pi_L^*) = \infty$.

According to Lemma 6, we have

$$\begin{aligned} S_{l+1} &= \max\left\{\left(\overline{b}_0 - 1\right) t\,(l+1), S_l + p\right\} \quad \text{if } l \le h-1 \\ S_{l+1} &= \max\left\{\left(\overline{b}_0 - 1\right) th + \overline{b}_0 t\,(l+1-h), S_l + p\right\} \quad \text{if } l \ge h, \\ &\qquad l = 1, 2, \ldots, L-1. \end{aligned} \tag{21}$$

The associated objective value can be derived as

$$\begin{aligned} &F_2(\pi_L^*) \\ &= \min\left\{\sum_{l=1}^{h} m\left(\overline{b}_0 - 1\right)(S_l + p) + \sum_{l=h+1}^{L} m\overline{b}_0\,(S_l + p)\right\} \\ &\quad - g\,(S_L + p) + X(L). \end{aligned} \tag{22}$$

Thus, the objective value is

$$F(\pi_L^*) = \min\left\{F_1(\pi_L^*), F_2(\pi_L^*)\right\}. \tag{23}$$

Theorem 12. *The problem $Vm \to batch|p_j = p, t_j = t, w_j|\sum C_j + X(x)$ can be solved by Algorithm D2 in $O(n/m)$ time.*

Proof. Equations (18) and (21) present that the machine can start to process B_{l+1} as soon as the orders assigned into B_{l+1} have arrived at the machine and the machine has finished processing B_l. The local optimal solution in two cases can be obtained by (19) and (22). The optimal solution is derived as (23). Hence, the optimal schedule to this special case can be obtained by repeating the above procedure for $n_0' \le L \le n_0$ and selecting the best candidate among the solutions generated. This shows that Algorithm D2 can find optimal solution. It is clear that the overall time complexity of Algorithm D2 is $O(n/m)$. □

We now demonstrate the above algorithm with the following numerical example.

Example 13. Consider the instance with 11 orders, $m = 2$, $t = 1$, $c = 4$, $W = 2$, $p = 2$, and $X(x) = 5x$. Based on the above method, we know $n_0 = 6$, $g = 1$, and $n_0' = 2$. Then, $L = 2, 3, 4, 5, 6$. We have the following results:

when $L = 2$, we have $\overline{b}_0 = 3$, $h = 0$, $S_1 = 3$, and $F(\pi^*) = 80$;

when $L = 3$, we have $\overline{b}_0 = 2$, $h = 0$, $S_1 = 2$, and $F(\pi^*) = 79$;

when $L = 4$, we have $\overline{b}_0 = 2$, $h = 2$, $S_1 = 1$, and $F(\pi_L^*) = 101$;

when $L = 5$, we have $\overline{b}_0 = 2, h = 4$, $\overline{b}_0 t \cdot 3 + t + W < S_3 + p$, and $F(\pi_L^*) = \infty$;

when $L = 6$, we have $\overline{b}_0 = 1, h = 0$, $\overline{b}_0 t \cdot 3 + t + W < S_3 + p$, and $F(\pi_L^*) = \infty$.

Therefore, we can obtain an optimal schedule π^* for $L = 3$ with $B_1 = \{J_1, J_2, J_3, J_4\}$, $B_2 = \{J_5, J_6, J_7, J_8\}$, and $B_3 = \{J_9, J_{10}, J_{11}\}$.

6. Concluding Remarks

In this paper, we address a coordinated scheduling problem with order transportation before processing and parallel-batch production under the limited waiting time consideration. Our goal is to optimize the sum of the total flow time and the total processing cost. First, we prove that the general problem is NP-hard in the strong sense. We also demonstrate that the problem with equal processing times on the machine is NP-hard. Furthermore, a dynamic programming algorithm in pseudopolynomial time is provided to prove its ordinarily NP-hardness. The problem with equal processing times and equal transportation times for each order can be solved in polynomial time through taking into account the properties of the special case. Our work has practical implications for the coordination of batch-production and transportation to improve the overall system performance in logistics and operations management. Simultaneously, another important implication of our paper may provide a vital approach to deal

with the waiting time constraint applied to reduce the raw material consumption and improve the machine utilization.

There are several possible extensions to this research. First, it is interesting to investigate the problems with other objective functions such as minimizing the makespan or minimizing maximum order tardiness/earliness. Another interesting issue is to develop effective heuristics to solve the general problem and investigate polynomial time algorithms for other special cases.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is partly supported by the National Natural Science Foundation of China (Grant no. 71101097) and the Program for Liaoning Excellent Talents in University (Grant no. LJQ2012017).

References

[1] P. Brucker, A. Gladky, H. Hoogeveen et al., "Scheduling a batching machine," *Journal of Scheduling*, vol. 1, no. 1, pp. 31–54, 1998.

[2] C. N. Potts and M. Y. Kovalyov, "Scheduling with batching: a review," *European Journal of Operational Research*, vol. 120, no. 2, pp. 228–249, 2000.

[3] C.-Y. Lee, R. Uzsoy, and L. A. Martin-Vega, "Efficient algorithms for scheduling semiconductor burn-in operations," *Operations Research*, vol. 40, no. 4, pp. 764–775, 1992.

[4] L. Tang, J. Guan, and G. Hu, "Steelmaking and refining coordinated scheduling problem with waiting time and transportation consideration," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 239–248, 2010.

[5] C.-Y. Lee and V. A. Strusevich, "Two-machine shop scheduling with an uncapacitated interstage transporter," *IIE Transactions*, vol. 37, no. 8, pp. 725–736, 2005.

[6] L. Tang and P. Liu, "Flowshop scheduling problems with transportation or deterioration between the batching and single machines," *Computers and Industrial Engineering*, vol. 56, no. 4, pp. 1289–1295, 2009.

[7] H. Gong and L. Tang, "Two-machine flowshop scheduling with intermediate transportation under job physical space consideration," *Computers and Operations Research*, vol. 38, no. 9, pp. 1267–1274, 2011.

[8] L. Tang and H. Gong, "A hybrid two-stage transportation and batch scheduling problem," *Applied Mathematical Modelling*, vol. 32, no. 12, pp. 2467–2479, 2008.

[9] C.-L. Li and J. Ou, "Machine scheduling with pickup and delivery," *Naval Research Logistics*, vol. 52, no. 7, pp. 617–630, 2005.

[10] Z.-L. Chen, "Integrated production and outbound distribution scheduling: Review and extensions," *Operations Research*, vol. 58, no. 1, pp. 130–148, 2010.

[11] S. Li, J. Yuan, and B. Fan, "Unbounded parallel-batch scheduling with family jobs and delivery coordination," *Information Processing Letters*, vol. 111, no. 12, pp. 575–582, 2011.

[12] L. Lu and J. Yuan, "Unbounded parallel batch scheduling with job delivery to minimize makespan," *Operations Research Letters*, vol. 36, no. 4, pp. 477–480, 2008.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide To the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY, USA, 1979.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 505207, 20 pages
http://dx.doi.org/10.1155/2014/505207

Research Article

Fuzzy Mixed Assembly Line Sequencing and Scheduling Optimization Model Using Multiobjective Dynamic Fuzzy GA

Farzad Tahriri,[1] Siti Zawiah Md Dawal,[1] and Zahari Taha[2]

[1] *Centre for Product Design and Manufacturing, Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia*
[2] *Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang Darul Makmur, Malaysia*

Correspondence should be addressed to Farzad Tahriri; farzad_tahriri@hotmail.com

Received 24 December 2013; Accepted 20 February 2014; Published 27 March 2014

Academic Editors: E. K. Aydoğan, J. G. Barbosa, and D. Oron

Copyright © 2014 Farzad Tahriri et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new multiobjective dynamic fuzzy genetic algorithm is applied to solve a fuzzy mixed-model assembly line sequencing problem in which the primary goals are to minimize the total make-span and minimize the setup number simultaneously. Trapezoidal fuzzy numbers are implemented for variables such as operation and travelling time in order to generate results with higher accuracy and representative of real-case data. An improved genetic algorithm called fuzzy adaptive genetic algorithm (FAGA) is proposed in order to solve this optimization model. In establishing the FAGA, five dynamic fuzzy parameter controllers are devised in which fuzzy expert experience controller (FEEC) is integrated with automatic learning dynamic fuzzy controller (ALDFC) technique. The enhanced algorithm dynamically adjusts the population size, number of generations, tournament candidate, crossover rate, and mutation rate compared with using fixed control parameters. The main idea is to improve the performance and effectiveness of existing GAs by dynamic adjustment and control of the five parameters. Verification and validation of the dynamic fuzzy GA are carried out by developing test-beds and testing using a multiobjective fuzzy mixed production assembly line sequencing optimization problem. The simulation results highlight that the performance and efficacy of the proposed novel optimization algorithm are more efficient than the performance of the standard genetic algorithm in mixed assembly line sequencing model.

1. Introduction

Mixed-model assembly lines (MMAL) have been widely used by manufacturers and they play a key role in the production of a variety of products. Products with similar characteristics are assembled with different processing times on the same assembly line at very low costs [1–3]. MMAL reduce setup operations to an extent that various models from a common base product can be manufactured in intermixed sequences. Mixed-model sequencing (MMS) aims at avoiding or minimizing sequence-dependent work overload based on detailed scheduling which explicitly accounts for operation time, worker movement, station borders, and other operational characteristics of the line [4]. MMS is an NP-hard problem which requires a unique and stable model to facilitate the production of mixed products in a manufacturing environment with multiple parts, machines, products, and assemblies in order to minimize scheduling time, idle time of the machines, setup number, and setup cost as well as maximize the number of products and assembly of various products [2, 3].

A contribution of this paper is to develop a new multiobjective fuzzy genetic algorithm model by integrating fuzzy expert experience controller (FEEC) with automatic learning dynamic fuzzy controller (ALDFC), which combines the execution time of GAs with dynamic control of population size, number of generations, tournament candidate, crossover rate, and mutation rate in order to solve a fuzzy mixed-model production line sequencing problem. The line-sequencing optimization model is based on two objectives such as minimizing a make-span and minimizing the machine setup number of the production line. The significance of this model is for those factories who want to produce various kinds of products with fixed machine just by changing the sequencing of the products. The model helps the manager to sequence and schedule the production line easily and accurately by taking the market demand into consideration.

2. Related Works

A number of studies attempted at solving multi- and mixed-product assembly lines sequencing problems using sequencing mathematical procedures and simulation models that will optimize various system measures such as throughput, scheduling time, number of stations, idle time, flow time, line length, work-in-process, and raw material demand deviations [1, 5–17].

In the advent of metaheuristic algorithms in recent years, numerous complex scheduling problems have been studied and solved using metaheuristic search techniques such as ant colony optimization (ACO), tabu search (TS), genetic algorithm (GA), and simulated annealing (SA). Metaheuristic algorithms are used to overcome the complexity of sequencing in assembly line problems [18].

Early research on the use of genetic algorithm (GA) for mixed-model line sequencing problems was carried out by Ghosh and Gagnon [19], whereby they introduced a mathematical programming model and an iterative GA-based procedure for MALBP with parallel work station. The goal of their research was to maximize the production rate of the line for a predetermined number of operators. A spreading and cutting sequencing (SCS) model using GA was implemented by Wong et al. [20] to solve the sequencing problem by reducing the completion time for daily operation of fabric spreading and cutting as well as improving the utilization of a computerized fabric cutting system used in the garment industry. Moon et al. [21] proposed an integrated machine tool selection and sequencing model to optimize the total production time and workload between machine tools using GA. In recent research, Norozi et al. [18] developed an intelligence-based GA in order to tackle the complexity of sequencing in parallel MMAL.

Most of the real-world decision problems involve multiple conflicting objectives that need to be tackled while adhering to the various constraints [22, 23]. Multiobjective optimization in mixed-model assembly line sequencing is dependent on the setup between product variants such as setup number, time, and cost as well as on the best sequence of total utility work to minimize the completion scheduling time (make-span) simultaneously. A number of studies used multiobjective optimization in mixed-model assembly line sequencing as follows: [2, 3, 24–30].

Genetic algorithm (GA) has been proven to be highly effective for achieving optimum or near-optimum solutions to complex real-world multiobjective optimization problems. However, GAs are limited by the fact that their performance is very sensitive to parameter settings. GA design consists of two key steps, namely, genetic operations and parameter settings [31]. The genetic operations involve choosing a suitable selection method. Parameter settings involve setting the required parameters and variables for controlling the algorithms such as population size, number of generations, number of selected candidates, crossover rate, and mutation rate [32]. Adaptive genetic operators (AGOs) are used in GA design for controlling a parameter. AGO can be classified into two modes, whereby the first mode involves implementing artificial intelligence techniques such as FLCs, and the second mode involves implementing conventional heuristics. The GA parameters controlled by these two modes are regulated adaptively during the genetic search process. This yields significant time savings during fine-tuning of the parameters, and the capabilities of GAs can be improved in searching for a global optimum [23, 33]. In adaptive genetic operators (AGOs) using fuzzy logic control, the fuzzy rules describing the relationship between the inputs and outputs need to be defined once the rule base has been determined. The rule base is determined after the inputs and outputs are selected and the database has been defined. There are various ways to achieve this objective and one way is to use the knowledge and experience of GA experts or by implementing an automatic learning technique for cases where knowledge and expertise are unavailable [34, 35]. Several works were focused on AGOs using artificial intelligent techniques for adaptive selection, crossover, and mutation in which the rules are based on the knowledge and experience of GA experts, as follows: [31, 32, 36]. A number of works on AGOs using artificial intelligent automatic learning technique rules include those of [35, 37–40]. Wang et al. [41] developed a fuzzy logic controlled genetic algorithm (FCGA) for environmental/economic dispatch. They proposed an improved genetic algorithm with two FLCs (one for crossover rate and mutation rate, resp.) based on several heuristics during the optimization process. The main concepts were implemented independently to adaptively regulate the crossover and mutation rates during the genetic search process. These parameters were taken as the input variables of the GAs, as well as the output variables of the FLCs.

It can be observed from the literature review that a few papers addressed mixed-assembly line sequencing and therefore there is a need for a detailed investigation on mixed-assembly line sequencing. It is found that the production of mixed-models is influenced by a number of criteria, which limit the applicability of the research results in real production line conditions. It shall be highlighted that only one criterion was considered in previous works such as operation line, and therefore other factors such as the travelling time of the conveyor were neglected. Deterministic timing has also been of interest in most studies. It shall be highlighted that only assembly line balancing was carried out in previous works in order to minimize the prediction time for input data, which include the travelling and processing times for each available job. Although the results from mixed-assembly line sequencing studies are applicable in real manufacturing environments, little is known on fuzzy mixed-model assembly line sequencing. Hence, it is evident that there is a lack of studies which implements the concept of fuzzy time in order to minimize the predicted time of input data. In general, the research papers can be classified into two groups, whereby the first group focuses solely on objective criteria, while the second group focuses on multiobjective investigations. A critical evaluation of previous works clarifies that addressing these objectives involves the development of various methods, in which MMAL sequencing is the ideal method for a single objective. Comparison of various GA methods reveals that multiobjective studies have not been investigated extensively, unlike single-objective studies.

Much effort has been made to intensify and accelerate the running of GA methods to achieve optimum results. Although mixed-model assembly line sequencing is of prime importance, there is a lack of studies which focus on this topic. In this paper, a hybrid method is proposed, in which mixed-model assembly line sequencing is integrated with the operating and travelling time in the form of fuzzy numbers of a multiobjective optimization problem. It can be observed from the existing literature that much effort has been devoted to intensify and accelerate the execution of GAs to attain optimum results. A number of works were focused on FGAs, in particular, the augmentation of GAs using fuzzy logic. Several works were focused on improving the performance of FGA methods. It shall be highlighted that these works primarily consider proliferating certain control parameters in GAs such as the identification of the population size, number of generations, tournament candidate, crossover rate, and mutation rate. Accelerating the speed of these parameters is carried out dynamically. The authors noted that there exists a gap in these works, whereby the mixed model assembly line sequencing is overlooked for each GA run. In light of the above review, this study is aimed at developing a new multiobjective fuzzy genetic algorithm by integrating fuzzy expert experience controller (FEEC) with automatic learning dynamic fuzzy controller (ALDFC), which combine the execution time of GAs with dynamic control of population size, number of generations, tournament candidate, crossover rate, and mutation rate in order to solve a fuzzy mixed-model assembly line sequencing problem.

3. Multiobjective Sequencing Problem in Mixed Model Assembly Line

The procedure of the proposed multiobjective fuzzy mixed assembly line sequencing model by integrating a fuzzy expert experience controller (FEEC) with automatic learning dynamic fuzzy controller (ALDFC) technique in genetic algorithm is shown in Figure 1. These steps are carried out for various types of applications and include input data, fuzzy variables, initialization, evaluation, selection, crossover, mutation, and termination. The input data are coded by considering the mixed model assembly line sequencing during the initialization of parameters. The operation time and travelling time are represented by fuzzy numbers. The fuzzy expert experience controller is used in order to determine the number of generations and population size in genetic algorithms. The fitness function values are grouped into two main objectives, that is, minimizing the make-span and setup number. The automatic learning dynamic fuzzy controller technique was adopted in order to dynamically control the genetic operators based on existing conditions such as selecting the number of tournament candidates as well as setting the crossover and mutation rates. Nine rules are used to control the crossover rate during the crossover stage, whereas eight rules are used to control the mutation rate during the mutation stage. Two rules are designed for the termination stage in order to control termination and gain optimum results.

The model is developed based on the works of [2, 24, 28, 42–44]. Moreover, the main references which support the development of the method in this research are [31, 33, 37, 45, 46].

3.1. Mixed Model Assembly Line Sequencing (MMALS). The input data involves identifying the number of machines (M_i) for producing the parts (P_i) or products ($\mathrm{P.NO}_i$) such as CNC, NC, and Robot as well as assigning the parts to their respective machines and robots based on the production and assembly line sequence ($\mathrm{A.S}_i$).

The processing time ($\widetilde{\mathrm{OP}}$) of each robot and machine as well as the travelling time ($\widetilde{\mathrm{T.t}}$) of each part assigned to the machines is represented as fuzzy numbers in the mixed-model assembly line sequencing model, rather than deterministic time values. The output data will become more accurate and representative of the real-case data by implementing trapezoidal fuzzy numbers (TFNs). The same procedure used by Fonseca et al. [47] is adopted in order to adapt the deterministic operation and travelling time in the fuzzy domain.

Initialization of parameters involves setting the parameters of the GA, creating the scores for the simulation, and creating the first generation of chromosomes. A total of 18 parameters are set during initialization, as shown in Table 1.

A general model for the mixed-model assembly line sequencing problem is shown in Figure 2, in which the parameters are listed in Table 1. The main solid arrows ($\rightarrow$) in Figure 2 represent the sequence to produce the parts, followed by a discrete part manufacturing assembly, leading to the final product. Likewise, the dashed lines (- - ->) represent the inputs to the same machine. The elapsed time between machines in order to manufacture a part is given by $(\widetilde{\mathrm{T.t}}j, j+1)$. The results of the model are dependent on the difficulties encountered during production planning and sequence. The number of genes in the chromosome (X_i) is formed based on the job numbers (J_i). Once the chromosomes have been formed, the chromosomes will be filled with random numbers via stochastic repeating, neglecting the encoding sequence of the produced chromosomes. The numbers vary between 0 and the maximum number of genes in the chromosomes (X_n). Once the chromosomes have been filled with random numbers, the classified gene code numbers are ranked into a possible sequence which can be used by the GA. The steps are described briefly as follows.

3.1.1. Identification of the First of Each Part Type's Gene Code. The first of each part type's gene code is identified using (1), whereby each part has a sequence of gene codes with k-elements $(M_1, M_2, \ldots, M_k)$. The values for $M_1, M_2, \ldots, M_k$ are given:

$$X_i - M_1 \leq (k-1)\,. \tag{1}$$

3.1.2. Classification Based on the Product's Part Sequence. Each gene code number in the chromosome line is compared to the previous one. Three conditions are used when comparing the gene code numbers, as described below.

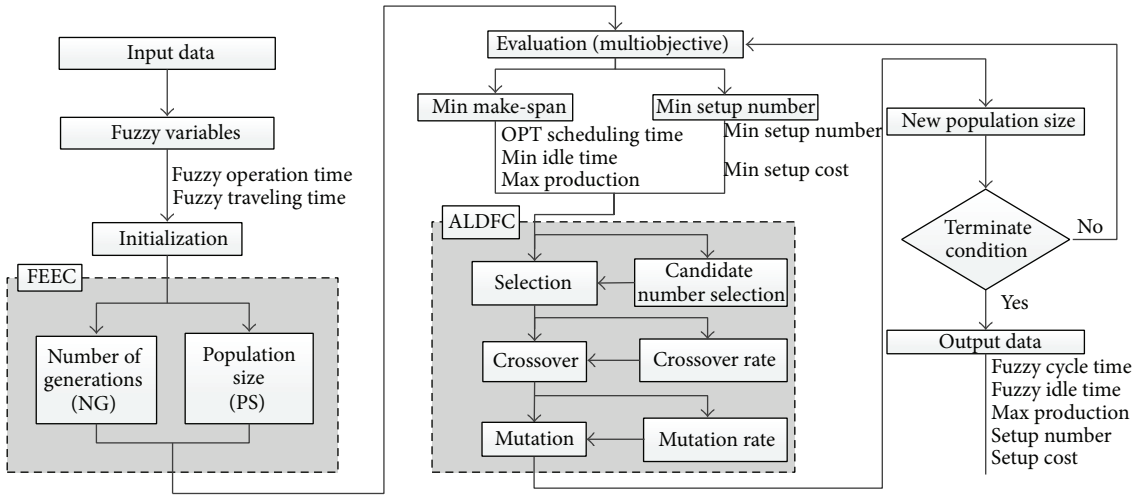


Figure 1: Flowchart of the proposed multiobjective fuzzy mixed assembly line sequencing model by using fuzzy genetic algorithm approach.

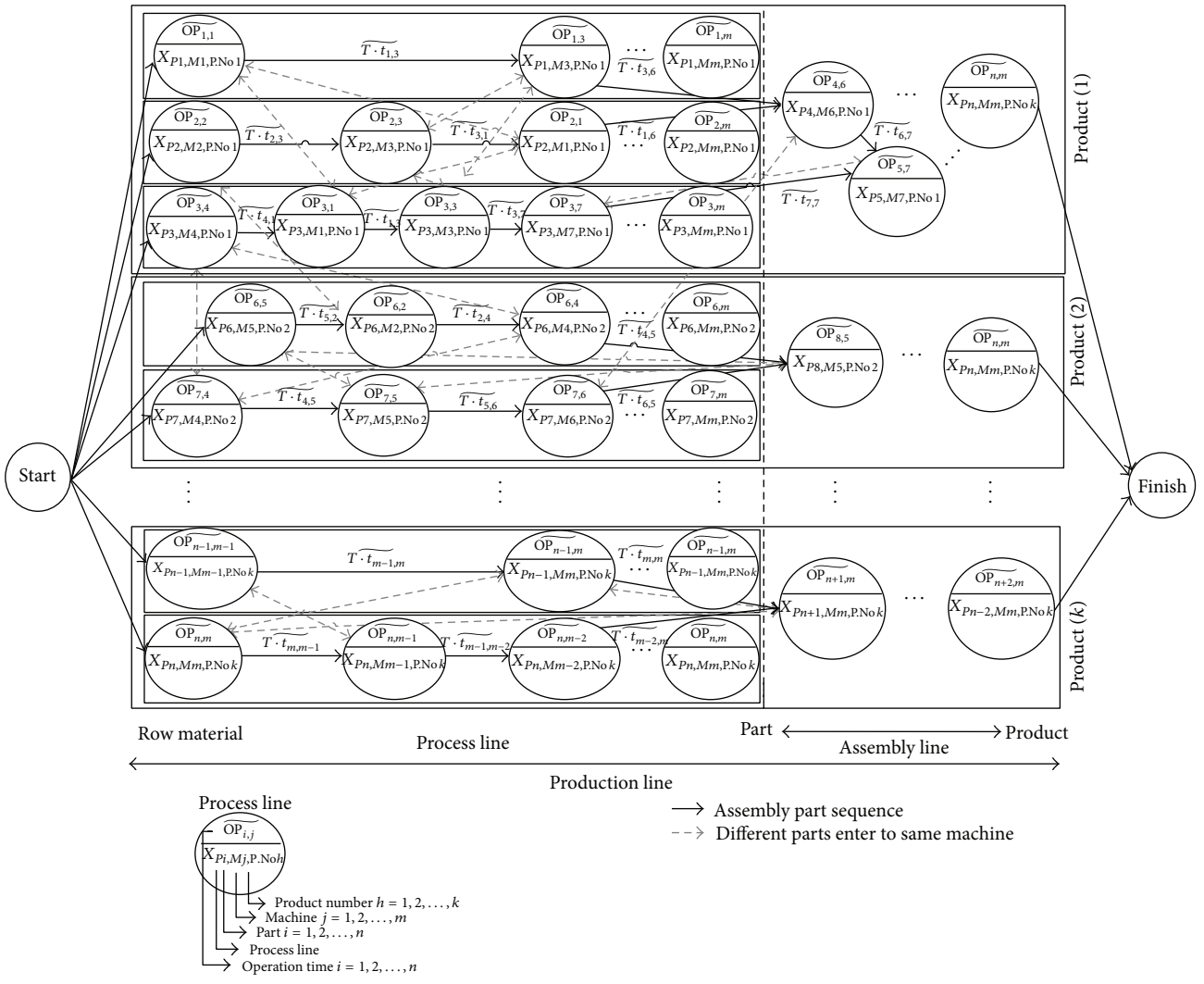


Figure 2: Example of a general mixed-model assembly line problem.

Table 1: Parameters used to develop the mixed-model assembly line sequencing model with genetic algorithm.

Gene code: $(X_i,\ i = 1, 2, \ldots, n)$	Machine number: $(M_i,\ i = 1, 2, \ldots, n)$	Tournament candidate: (TC)
Jobs: $(J_i,\ i = 1, 2, \ldots, n)$	Fuzzy operation time: $(\widetilde{\mathrm{OP}}(a_n, b_n, c_n))$	Crossover rate: (CR)
Product number: $(\mathrm{P.NO}_i, i = 1, 2, \ldots, n)$	Fuzzy traveling time: $(\widetilde{\mathrm{T.t}}(a_n, b_n, c_n))$	Mutation rate: (MR)
Assembly sequence: $(\mathrm{A.S}_i, i = 0, 1, \ldots, n)$	Elitism: $(E = 3)$	Fuzzy start time: $(\widetilde{\mathrm{S.t}})$
Setup number: (S.No = A, B, ..., Z)	Population size: (PS)	Fuzzy total scheduling time: $(\widetilde{C})$
Part: $(P_i,\ i = 1, 2, \ldots, n)$	Maximum generations: (Max$G$)	Chromosome: (Ω)

(i) The first or existing gene code number fills up the new chromosome without any changes in the gene code number.

(ii) If the gene code number is the same as another gene code number within the chromosome, the chromosome is filled up with the addition of one number out of the existing gene code numbers.

(iii) If the gene code number is different from other gene code numbers within the chromosome, the existing gene code number fills up the chromosome without any changes in the gene code number.

3.1.3. Classification Based on the Assembly Sequence. Once the product's part sequence has been classified, the chromosome is filled up using the assembly sequence number related to a gene code (X_i) of the specific chromosome. Finally, the final gene code numbers are ranked by two sequence filters, which accounts for the assembly sequence code and product's part sequence.

3.2. Objective Functions. The fitness function values for multiobjective mixed-production assembly line sequencing are categorized into two main objectives, that is, minimizing the total make-span and setup number. Each chromosome is evaluated during each generation of the selection process. This is accomplished by looking up the score of each gene in the chromosome, adding the scores, and averaging the scores for the chromosome. The elite chromosome of the generation is determined as part of the evaluation process. The following assumptions are made for the multiobjective evaluation.

(1) The conveyor (operator movement) moves at a constant speed. If job overlapping occurs, the remaining work will be accomplished by temporary operators.

(2) The job operation begins when the part enters the machine. Once the job is completed, the operator will move the part to the next machine.

(3) The operator is assigned to each selected part that is assigned to each machine.

(4) The position of the machines on the assembly line varies from one to another depending on the user input, which is based on the travelling time.

(5) The assembly line can process jobs for a product family, which is described by a joint priority matrix.

(6) The processing time varies for different jobs and these jobs are allocated to the same machine. However, what is the optimum processing time for each job?

(7) The demand for all products and the sequence of the products entering the assembly line are predetermined.

(8) The completion time for all jobs is represented by a fuzzy number.

3.2.1. Minimizing the Total Make Span. The step involves evaluating each chromosome based on the make-span by considering the product number and assembly sequence for a gene code (X_i). The x_1, x_2, x_3, and S.t(X_i) are computed in which X_i is calculated as follows. x_1 is determined by checking the start time of the parts entering the machines, based on the sequence assigned to the machines. This is expressed by

$$x_1 = \begin{cases} 0, & P(X_i) \neq P(X_k), \\ \mathrm{S.t}(X_k) + \mathrm{OP}(X_k) + \mathrm{T.t}(X_k), & P(X_i) = P(X_k), \end{cases} \quad k = i-1,\ i-2, \ldots, 2, 1, \tag{2}$$

where X_2 is determined by checking the start time of the parts entering the machines based on the assigned machines, as given by

$$x_2 = \begin{cases} 0, & M(X_i) \neq M(X_k), \\ \mathrm{S.t}(X_k) + \mathrm{OP}(X_k), & M(X_i) = M(X_k), \end{cases} \quad k = i-1, i-2, \ldots, 2, 1, \tag{3}$$

where x_3 is then determined by checking the start time of the parts entering the machines based on the sequence of the part's assembly for each product, as given by (6) and (7).

For each X_i where $i = 1, 2, \ldots, n$, note that the product number (P.No(X_i)) and assembly sequence (A.S(X_i)) are important. Equations (4) through (7) are given below.

(i)

$$\text{If } \mathrm{A.S}(X_i) = 0 \Longrightarrow x_3 = 0, \quad i = 1, 2, \ldots, n, \tag{4}$$

$$\text{If } \mathrm{A.S}(X_i) \neq 0 \Longrightarrow \text{compares } \mathrm{P.No}(X_i),\ \mathrm{P.No}(X_{i-1}). \tag{5}$$

(ii) If $\text{P.No}(X_i) = \text{P.No}(X_{i-1})$, then

$$x_3 = \begin{cases} \text{S.t}\left(X_{i-1}\right), \\ \quad \text{A.S}\left(X_i\right) = \text{A.S}\left(X_{i-1}\right), \\ \text{S.t}\left(X_{i-1}\right) + \text{OP}\left(X_{i-1}\right) + \text{T.t}\left(X_{i-1}\right), \\ \quad \text{A.S}\left(X_i\right) \neq \text{A.S}\left(X_{i-1}\right), \end{cases} \quad (6)$$

$$i = 1, 2, \ldots, n.$$

(iii) If $\text{P.No}(X_i) \neq \text{P.No}(X_{i-1})$ when searching a gene code X_k, where $k = 1, 2, \ldots, i-2$, go backwards one by one until $\text{P.No}(X_i) = \text{P.No}(X_k)$ is met:

$$x_3 = \begin{cases} \text{S.t}\left(X_k\right), \\ \quad \text{A.S}\left(X_i\right) = \text{A.S}\left(X_k\right), \\ \text{S.t}\left(X_k\right) + \text{OP}\left(X_k\right) + \text{T.t}\left(X_k\right), \\ \quad \text{A.S}\left(X_i\right) \neq \text{A.S}\left(X_k\right), \end{cases} \quad (7)$$

$$i = 1, 2, \ldots, n.$$

The start time is obtained using (8), in which x_1, x_2, x_3 are determined from the previous steps:

$$\text{S.t}\left(X_i\right) = \text{Max}\left(x_1, x_2, x_3\right); \quad X_i,\ i = 1, 2, \ldots, n. \quad (8)$$

Note. In the first run, if $x_1 = x_2 = x_3 = 0$, then S.t = Max(x_1, x_2, x_3) + T.t.

The make-span fitness function is then calculated for all chromosomes using

$$\text{Final scheduling time} = \text{Max S.t}\left(X_i\right) + \text{OP}\left(X_i\right), \quad i = 1, 2, \ldots, n. \quad (9)$$

3.2.2. Minimizing the Setup Number. The machine number $M(X_i)$ and setup number $\text{S.No}(X_i)$ are defined for element (X_i) in the chromosome, where $i = 1, 2, \ldots, n$, once $M(X_i)$ and $S.\text{No}(X_i)$ have been determined for all X_i (where $i = 1, 2, \ldots, n$). If $M(X_i) = M(X_j) = t$ (where $i, j = 1, 2, \ldots, n$ and $i \neq j$, $t = 0, 1, \ldots, n$), then the sequence is $m_1, m_2, \ldots, m_n$ where $(m_k, k = 1, 2, \ldots, n)$ are the setup numbers of the same machine. Finally, (m_k, m_{k+1}) are compared using the following equations (where $k = 1, 2, \ldots, n-1$):

$$P^t_{(k,k+1)} = \begin{cases} 1, & m_k \neq m_{k+1}, \\ 0, & m_k = m_{k+1}. \end{cases} \quad (10)$$

The total setup number for each machine is

$$t = \sum_{k=1}^{n} P^t_{(k,k+1)} + 1. \quad (11)$$

The evaluation setup number (E.S.N) is determined for all machines using

$$\text{E.S.N} = \sum_{t=0}^{n} \left(\sum_{k=1}^{n} P^t_{(k,k+1)} + 1 \right). \quad (12)$$

Finally, the total fitness values of the efficient frontiers are calculated based on these two objectives [27]. This step is repeated for each possible chromosome (X_i) in the population size.

3.3. Adaptive Genetic Operators

3.3.1. Fuzzy Expert Experience Controller. Fuzzy expert experience controller (FEEC) is implemented to set the population size, number of generations, and other variables prior to execution of the GA. The FEEC is used to control the parameters automatically. The controller comprises four principal components, listed as follows:

(1) fuzzification interface, which converts crisp input data into suitable linguistic values;

(2) fuzzy rule base, which consists of a set of linguistic control rules incorporating heuristics that are used to achieve a faster rate of convergence;

(3) fuzzy inference engine, which is a decision-making logic that employs rules of the fuzzy rule base to infer fuzzy control actions in response to fuzzed inputs;

(4) defuzzificationinterface, which yields a crisp control action from an inferred fuzzy control action.

The first step (fuzzification) involves defining the membership function of inputs and outputs for each parameter. The properties of these attributes (i.e., fuzzy variables, fuzzy set and fuzzy number of each input, and output variable) are listed in Table 2. The membership functions of the fuzzy input and output variables for each aspect are formulated based on the knowledge and experience of GA experts, as well as related literature [32, 36]. The membership functions are defined by the population size, ranging from 10 to 160 in the experiments. The membership function variables are all trapeziums, ranging from 0 to 1 on the defined universe of discourse. The inputs and outputs and their associated sets of linguistic labels are illustrated in Table 2. The input variable is chromosome size (CS) and is divided into four trapezoid membership functions, namely, "very short (VSH)", "short (SH)", "long (LO)," and "very long (VL)." The membership values assigned to the two output variables are "population size (PS)" and "number of generations (NG)." Four linguistic labels, namely, "small (S)," "medium (M)," "big (B)," and "very big (VB)" are used to represent Output 1 (i.e., population size (PS)) in which the universe of discourse has values within the range of 0 and 200. Likewise, the same four linguistic labels are used to represent output 2 (i.e., number of generations) whereby the universe of discourse consists of values ranging between 0 and 1750.

The second step involves formulating the fuzzy rules for each trapezoidal membership function. The number of rules (N) required to control the system is given by

$$N = \sum_{j=1}^{m} \left(\prod_{i=1}^{n} L_i \right), \quad (13)$$

where m represents the number of sets of rules, L_i represents the number of membership functions or levels, and N represents the number of input variables used in one set of rules. If $M = 1$, $N = 1$, and $L_i = 4$, the number of rules (N) will be $1 \times 1 \times 4 = 4$. The relationship between the fuzzy input and fuzzy output variables for each station is developed using the Mamdani fuzzy IF-THEN MIN-MAX rules. The fuzzy

Table 2: List of properties of attributes.

Fuzzy variables	Fuzzy set	Fuzzy number
Fuzzy input variables		
Chromosome size (CS)	VSH, SH, LO, VL	(0, 0, 10, 20) (10, 15, 25, 30) (20, 30, 50, 70) (50, 70, 130, 130)
Fuzzy output variables		
(i) Population size (PS)	S, M, B, VB	(0, 20, 60, 80) (60, 70, 90, 100) (80, 95, 105, 120) (110, 130, 200, 200)
(ii) Number of generations (NG)	S, M, B, VB	(0, 50, 150, 200) (150, 200, 400, 450) (400, 450, 1150, 1200) (110, 1200, 1700, 1750)

Table 3: Fuzzy IF-THEN rules for chromosome size (CS) input variable.

Rule number	Fuzzy input variable	Fuzzy output variables	
	Chromosome size (CS)	Population size (PS)	Number of generations (NG)
1	Very short (VSH)	Small (S)	Small (S)
2	Short (SH)	Medium (M)	Medium (M)
3	Long (LO)	Big (B)	Big (B)
4	Very long (VL)	Very big (VB)	Very big (VB)

rules are developed based on interviews with experienced experts. The collection of fuzzy rules approximately represents the human thinking process during decision making. In single-input multiple-output (SIMO) systems, these rules are considered to be heuristic design rules of the following form [48]:

IF input (1) is "A," then output (1) is "B" and output (2) is "C."

A sample of the generated fuzzy rules for chromosome size (CS) input variable is presented in Table 3.

The basic fuzzy rule-based system used for the fuzzy population and generation size model is summarized in Table 4. It can be seen from Table 4 that the Mamdani fuzzy inference model is used as the conjunction operator and is based on the aggregation function maximum (Max).

The third step involves designing fuzzy implication (FI) for each rule. The rules' weights need to be determined prior to FI. The input of the implication process is a single number given by the antecedent, and the output is a fuzzy set. Aggregation is equivalent to fuzzification, when there is only one input to the controller. The output decisions are based on testing all rules in the FEEC, and the rules (R_i) must be combined in a specific manner for decision making. The aggregation operation is used during the calculation of the degree of aggregation (a_i) for the condition of a rule (R_i). Aggregation is a process by which the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule.

After implication, the output decisions following testing of all rules in the FEEC and the rules (R_i) must also be combined in a specific manner for decision making. The rules are placed to show how the output of each rule is combined or aggregated into a single fuzzy set whose membership function assigns a weight for every output (tip) value. Finally, these fuzzy outputs need to be converted into a scalar output quantity so that the nature of the action to be performed can be determined by the centre of gravity method. The most popular defuzzification method is the centroid calculation method, which returns the centre of gravity (COG) under the curve. This technique was developed by Sugeno in 1985 and is the most commonly used technique as well as being highly accurate. The centered defuzzification technique can be expressed as

$$X^* = \frac{\int \mu_i(x)\, x\, dx}{\int \mu_i(x)\, dx}, \tag{14}$$

where X^* is the defuzzified output, $\mu_i(x)$ is the aggregated membership function, and x is the output variable. The only disadvantage of this method is that it is computationally difficult for complex membership functions.

3.3.2. Fuzzy Automatic Learning Controller. The fuzzy automatic learning controller (FALC) technique is implemented to adjust the GA operators (e.g., tournament candidate selection and crossover and mutation rates) automatically during the optimization process. The heuristic updating principles of the tournament candidate and crossover and mutation rates are such that the change in the average fitness of the population is greater than zero and remains the same sign in consecutive generations. Put simply, the GA operators will increase and vice versa. The block diagram of the proposed fuzzy logic controlled genetic algorithm (FLCGA) for tournament candidate and crossover and mutation rates is shown in Figure 3. The tournament candidate and crossover and mutation rates can be adjusted adaptively or dynamically during the evolution process due to the fuzzy logic embedded in the genetic operators.

From Figure 3, the online FLCs are used to adapt the tournament candidate and crossover and mutation rates with the aim of improving the convergence rate significantly. The FLCs shown in Figure 3 are used to control the parameters automatically and comprise four principal components, which are listed as follows:

(1) fuzzification interface, which converts crisp input data into suitable linguistic values;

Table 4: Summary of the basic fuzzy rule-based system.

Aspect number	Aspect name	Number of inputs	Number of outputs	Number of rules	Aggregation operator	Inference model
1	Population and generation size	1	2	4	Max	Mamdani

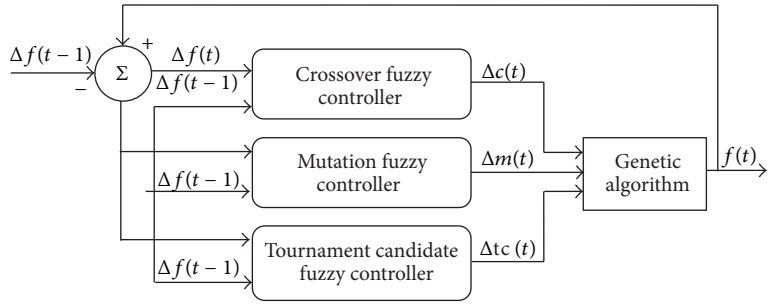


Figure 3: Block diagram of proposed fuzzy controlled genetic algorithm for tournament candidate and crossover and mutation rates.

(2) fuzzy rule base, which consists of a set of linguistic control rules incorporating heuristics used to achieve a faster convergence rate;

(3) fuzzy inference engine, which is a decision making logic that employs rules from the fuzzy rule base to infer fuzzy control actions in response to the fuzzed inputs;

(4) defuzzificationinterface, which yields a crisp control action from an inferred fuzzy control action.

The fundamental concept of Song et al. [37] is implemented to regulate the GA operators (tournament candidate "$\Delta tc(t)$," crossover rate "$\Delta c(t)$," and mutation rate "$\Delta m(t)$") adaptively using FLC during the genetic search process. The heuristic updating strategy for these three operators is based on the changes in the average fitness of the GA population for two continuous generations, "$\Delta f(t)$" and "$\Delta f(t-1)$." The number of tournament candidates (TC), crossover rates (C), mutation rates (M), and occurrence rates of the operators will increase if they consistently produce better offspring during the recombination process. Likewise, the number of TC, C, M, and occurrence rate of the operators will decrease if they consistently produce poorer offspring. This scheme is based on the principle that well-performing operators are encouraged to produce a higher number of offspring, while reducing the chance for poor-performing operators to destroy the potential individuals during the recombination process. The inputs of the fuzzy tournament candidate and crossover and mutation rates are changes in fitness at two consecutive steps (i.e., $\Delta f_{\text{avg}}(t-1)$ and $\Delta f_{\text{avg}}(t)$) while the outputs are the changes in the number of tournament candidates ($\Delta tc(t)$) and crossover rates ($\Delta c(t)$) and mutation rates "$\Delta m(t)$." The change in average fitness at generation (t) for the minimization problem (i.e., $\Delta f_{\text{avg}}(t)$) is set using

$$\begin{aligned}\Delta f_{\text{avg}}(t) &= \left(\overline{f_{\text{par-size}}}(t) - \overline{f_{\text{off-size}}}(t)\right) \times \lambda \\ &= \left(\frac{\sum_{k=1}^{\text{par-size}} f_k(t)}{\text{par} - \text{size}} - \frac{\sum_{k=\text{par-size}+1}^{\text{par-size+off-size}} f_k(t)}{\text{off} - \text{size}}\right) \times \lambda,\end{aligned} \tag{15}$$

where k is a scaling factor to normalize the average fitness values for defuzzification in the FLC and is varied accordingly to the given problem. The scaling factor k is required to normalize the average fitness values. The regulation and procedure of $\Delta tc(t)$, $\Delta c(t)$, and $\Delta m(t)$ begin with the application of $\Delta f_{\text{avg}}(t-1)$ and $\Delta f_{\text{avg}}(t)$ based on the average fitness values, as in Algorithm 1, where ε is a given real number in the proximity of zero and γ and $-\gamma$ represent the given maximum and minimum values, respectively, for the fuzzy membership function of the fuzzy input and output linguistic variables, as illustrated in Figure 4. The labels of the membership function are as follows: NL: negative larger, NR: negative large, NM: negative medium, NS: negative small, ZE: zero, PS: positive small, PM: positive medium, PR: positive large, PL: positive larger, TW: two, TH: three, FO: four, and FI: five.

The $\Delta f_{\text{avg}}(t-1)$ and $\Delta f_{\text{avg}}(t)$ values are normalized correspondingly within the range of $[-1.0, 1.0]$. The $\Delta tc(t)$ values are normalized within the range of $[2, 5]$, whereas the $\Delta c(t)$ and $\Delta m(t)$ values are normalized within the range of $[-0.1, 0.1]$ and $[-0.01, 0.01]$, respectively, depending on their corresponding maximum values.

The application of a new tournament candidate fuzzy decision table designed based on conventional design concepts as well as the fuzzy decision table for crossover and mutation rates is given in Table 5.

For simplicity, a look-up table for the control actions of the tournament candidate and crossover rate and mutation rate FLCs is shown in Table 6. The quantified levels corresponding to the linguistic values of input and output fuzzy variables of the tournament candidate and crossover and

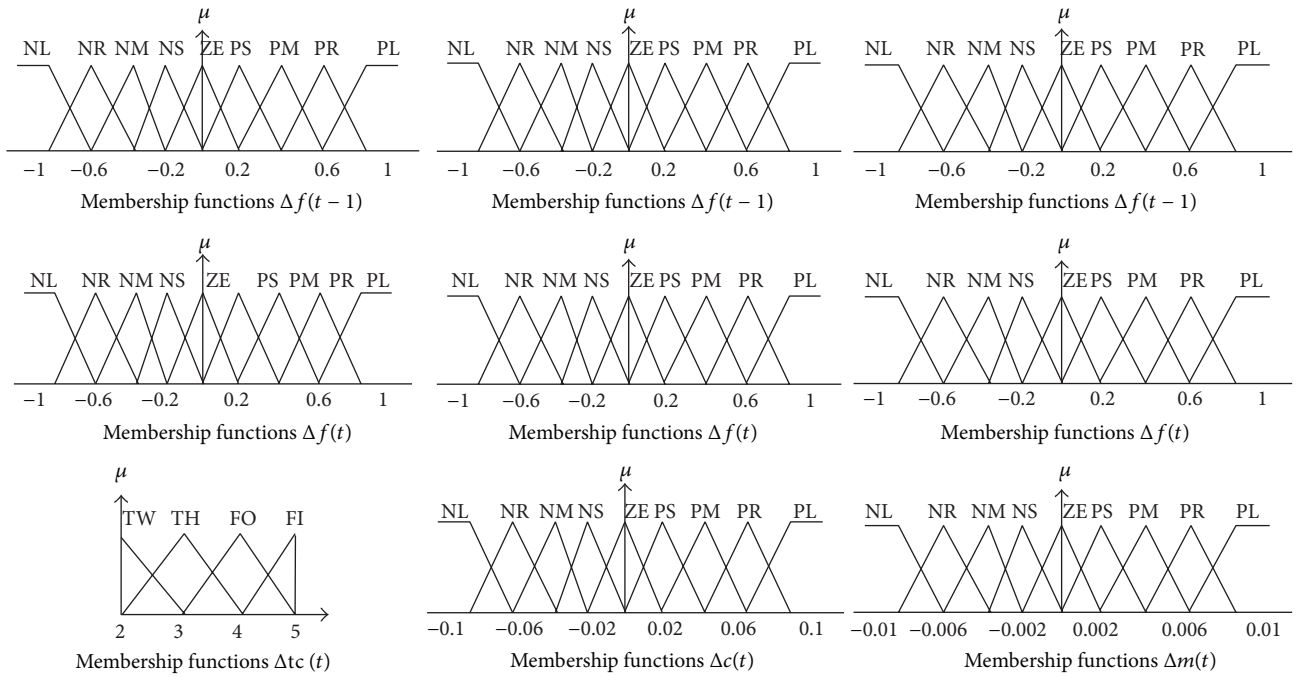


Figure 4: Membership functions of inputs ($\Delta f(t-1)$ and $\Delta f(t)$) and outputs ($\Delta tc(t), \Delta c(t)$, and $\Delta m(t)$).

If $\varepsilon \le \Delta f_{\text{avg}}(t-1) \le \gamma$ and $\varepsilon \le \Delta f_{\text{avg}}(t) \le \gamma$ then increase $N(\text{tc}, c, m)$ for next generation;
If $-\gamma \le \Delta f_{\text{avg}}(t-1) \le -\varepsilon$ and $-\gamma \le \Delta f_{\text{avg}}(t) \le -\varepsilon$
then decrease $N(\text{tc}, c, m)$ for next generation;
If $-\varepsilon \le \Delta f_{\text{avg}}(t-1) \le \varepsilon$ and $-\varepsilon \le \Delta f_{\text{avg}}(t) \le \varepsilon$
then rapidly increase $N(\text{tc}, c, m)$ for next generation;
End
End

Algorithm 1

mutation rates FLCs are designated as −4, −3, −2, −1, 0, 1, 2, 3, and 4, respectively. In order to calculate the crossover and mutation rates based on (2), let x represent the quantified levels of $\Delta f(t-1)$, y the quantified levels of $\Delta f(t)$, and z the quantified levels of $\Delta c(t)$ and $\Delta m(t)$. Equation (16), which is illustrated in Table 6, is given by

$$z \le \alpha x + (1-\alpha)\, y, \tag{16}$$

where x and y represent the first and second inputs of the average fitness functions ($\Delta f(t-1)$) and $\Delta f(t)$), respectively, whereas z is a minimum integer which is less than $ax + (1-\alpha)y$, and a is an adaptive coefficient which varies with the fitness value of the whole population. It is found that the crossover and mutation rates FLCs yield good performance when α equals 0.5.

Fuzzy inference engine, which is a decision making logic, employs the rules of the fuzzy rule base to infer fuzzy control actions in order to generate fuzzy outputs based on the inputs. The input values are assigned to the indices x and y upon identification of the fitness function values ($\Delta f_{\text{avg}}(t-1)$ and $\Delta f_{\text{avg}}(t)$). The input values correspond to the controller

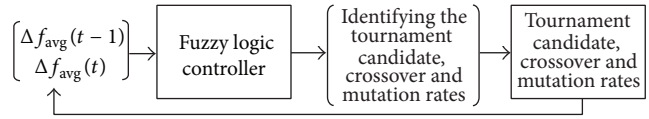


Figure 5: Regulating strategy of the fuzzy logic controller in the genetic algorithm loop.

actions based on the fuzzy decision table in order to identify the values of the tournament candidate, crossover rate, and mutation rate, as shown in Table 5. For fuzzy inference, the regulating strategy of the FLCs in the GA loop is shown in Figure 5.

The outputs of the tournament candidate ($\Delta tc(t)$) and changes in the FLC for crossover and mutation rates ($\Delta c(t)$ and $\Delta m(t)$) are generated upon identification of $Z(x, y)$ for the tournament candidate and crossover and mutation rates using Table 6 and (17), (18), and (19), respectively:

$$\Delta tc\,(t) = Z\,(x, y)\,, \tag{17}$$

Table 5: Fuzzy decision table for tournament candidate and crossover and mutation rates.

(a) $\Delta tc(t)$

$\Delta f(t)$	$\Delta f(t-1)$								
	NL	NR	NM	NS	ZE	PS	PM	PR	PL
NL	FI	FI	FI	FO	FO	TH	TH	TW	TW
NR	FI	FI	FO	FO	TH	TH	TW	TW	TH
NM	FI	FO	FO	TH	TH	TW	TW	TH	TH
NS	FO	FO	TH	TH	TW	TW	TH	TH	FO
ZE	FO	TH	TH	TW	FO	TH	TH	FO	FO
PS	TH	TH	TW	TW	TH	TH	FO	FO	FI
PM	TH	TW	TW	TH	TH	FO	FO	FI	FI
PR	TW	TW	TH	TH	FO	FO	FI	FI	FI
PL	TW	TH	TH	FO	FO	FI	FI	FI	FI

(b) $\Delta c(t)$ and $\Delta m(t)$

$\Delta f(t)$	$\Delta f(t-1)$								
	NL	NR	NM	NS	ZE	PS	PM	PR	PL
NL	NL	NR	NR	NM	NM	NS	NS	ZE	ZE
NR	NR	NR	NM	NM	NS	NS	ZE	ZE	PS
NM	NR	NM	NM	NS	NS	ZE	ZE	PS	PS
NS	NM	NM	NS	NS	ZE	ZE	PS	PS	PM
ZE	NM	NS	NS	ZE	ZE	PS	PS	PM	PM
PS	NS	NS	ZE	ZE	PS	PS	PM	PM	PR
PM	NS	ZE	ZE	PS	PS	PM	PM	PR	PR
PR	ZE	ZE	PS	PS	PM	PM	PR	PR	PL
PL	ZE	PS	PS	PM	PM	PR	PR	PL	PL

Table 6: Look-up table for control actions by tournament candidate, crossover rate, and mutation rate FLCs.

(a) $\Delta tc(t)$

$Z(X,Y)$		X								
		−4	−3	−2	−1	0	1	2	3	4
	−4	5	5	5	4	4	3	3	2	2
	−3	5	5	4	4	3	3	2	2	3
	−2	5	4	4	3	3	2	2	3	3
	−1	4	4	3	3	2	2	3	3	4
Y	0	4	3	3	2	4	3	3	4	4
	1	3	3	2	2	3	3	4	4	5
	2	3	2	2	3	3	4	4	5	5
	3	2	2	3	3	4	4	5	5	5
	4	2	3	3	4	4	5	5	5	5

(b) $\Delta c(t)$ and $\Delta m(t)$

$Z(X,Y)$		X								
		−4	−3	−2	−1	0	1	2	3	4
	−4	−4	−3	−3	−2	−2	−1	−1	−0	+0
	−3	−3	−3	−2	−2	−1	−1	−0	+0	1
	−2	−3	−2	−2	−1	−1	−0	+0	1	1
	−1	−2	−2	−1	−1	−0	+0	1	1	2
Y	0	−2	−1	−1	−0	+0	1	1	2	2
	1	−1	−1	−0	+0	1	1	2	2	3
	2	−1	−0	+0	1	1	2	2	3	3
	3	−0	+0	1	1	2	2	3	3	4
	4	+0	1	1	2	2	3	3	4	4

$$\Delta c(t) = Z(x, y) \times 0.02, \tag{18}$$

$$\Delta m(t) = Z(x, y) \times 0.002, \tag{19}$$

where $Z(x, y)$ consists of the corresponding values of $\Delta f_{\text{avg}}(t-1)$ and $\Delta f_{\text{avg}}(t)$ for defuzzification from Table 6, in which $x, y \in \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$. The values 0.02 and 0.002 are selected in order to regulate the increasing and decreasing range of rates for the crossover and mutation operators, respectively. The tournament candidate is computed using

$$tc(t) = \Delta tc(t). \tag{20}$$

The rate of the crossover and mutation operator is updated using (21) and (22), respectively, once the changes of the crossover rate ($\Delta c(t)$) and mutation rate ($\Delta m(t)$) have been identified using (5) and (6):

$$P_c(t) = P_c(t-1) + \Delta c(t), \tag{21}$$

$$P_m(t) = P_m(t-1) + \Delta m(t). \tag{22}$$

The adjusted rates should not exceed the range of 0.5–1.0 and 0.0–0.1 for $P_c(t)$ and $P_m(t)$, respectively [33].

Tournament selection is carried out once the adaptive genetic operators (i.e., number of selected candidates and crossover and mutation rates) have been identified. Tournament selection is a method used to reproduce a new generation which is proportional to the fitness of each individual. The main characteristics of tournament selection are summarized as follows.

(i) Tournament selection is quite useful in certain situations, such as multiobjective optimization.

(ii) Tournament selection uses only local information.

(iii) Tournament selection is easily implemented with low time complexity.

(iv) Tournament selection can be easily implemented in a parallel environment.

However, tournament selection also suffers from selection bias, which means that the best one will not be selected if it is very unlucky.

Following this, crossover probability is used, which crosses over parents to form new offspring (children). In the crossover phase, all chromosomes (except for the elite chromosome) are paired up and crossed over with a probability crossover rate. Crossover is accomplished by choosing a site randomly along the length of the chromosome and exchanging the genes of two chromosomes (parents) for each gene past this crossover site. The details for crossover are given as follows.

(1) Identify the number of different requirements for manufactured products (P. No).

(2) Create the one random number between product type numbers (P. No):

```
for (int i = 0; i < P.No; i++)
{
    Product Random = rand( ) % Product;
}
```

(3) Identify the gene code number (X_i) for the selected product obtained from the previous step.

(4) Search and identify the gene code from parent A based on random product selection and transfer these gene codes to child B, which is exactly at the same gene location.

(5) Search and identify the gene code from parent B based on random product selection and transfer the gene code to child A, which is exactly at the same gene location.

(6) Transfer the remaining gene code from parent A to the gene blank of child A.

(7) Transfer the remaining gene code from parent B to the gene blank of child B.

(8) Calculate the number of crossovers based on the crossover rate ($P_c(t)$) from (21) and population size (PS):

$$\text{Number of crossovers} = \frac{P_c(t) \times \text{PS}}{2}. \quad (23)$$

(9) Once new offsprings have been created, the new offspring will have previous chromosomes in the current generation.

In crossover operation, the worst or weakest chromosomes will fade away whereas the characteristics of the chromosomes will change continuously during mutation operation. The elite chromosome will not be subjected to mutation in the next generation. Consequently, GA does not lead to annihilation since several chromosomes (one, two, or three) from each generation are transferred directly to the next generation. Mutation is not applied on chromosomes which are immune. It is possible to maintain a fixed fitness value in some generations, but they will never deteriorate. The number of elitism is assigned as three.

Following crossover operation, the genes will mutate to any one of the codes at a mutation rate for each gene in the chromosomes, with the exception of the elite chromosome. When the crossover and mutation operations are complete, the chromosomes will be evaluated for another round of selection and reproduction. Considering elitism and after identifying the parts in which mutation will be applied, the number of mutations in each generation is calculated using (24) based on the mutation rate ($P_m(t)$), population size (PS), and maximum gene code (Max.X_i):

$$\text{Number of mutations} \cong \left[(\text{PS} \times \text{Max.X}_i) \times P_m\right]. \quad (24)$$

The swap mutation operator is used in this study, in which two positions are selected at random and their contents are swapped as a general procedure. After identifying the number of gene mutations, a set of rules need to be devised for the mutation of genes from point A to B and vice versa, while focusing on the stability of the chromosome sequence. It is important that the sequence of the chromosomes is not displaced. There are eight sets of rules for this step which are classified into two groups, as follows.

(1) Four rules are used to check the mutation based on the part sequence (Rules 1 and 2) and product assembly (Rules 3 and 4) of genes from A to B.

(2) Four rules are used to check the mutation based on the part sequence (Rules 5 and 6) and product assembly (Rules 7 and 8) of genes from B to A.

It shall be noted that the position of A is before B. The eight mutation rules should be checked thoroughly to ensure their ideal applications. In the adverse case, the eight rules ought to be done over.

Having performed crossover, elitism and mutation operations, the most ideal chromosomes of the current generation, are compared and evaluated to identify its total value, after checking its termination in the following step. The loop of chromosome generations is terminated when certain conditions are met. The elite chromosome is returned as the best solution once the termination criteria are met. The termination criteria are listed below.

(1) If the number of generations reaches its maximum, the loop of chromosome generations is terminated.

(2) If there are no changes in the elite solution (i.e., no changes in fitness function value), the loop of chromosome generations is terminated using

$$\text{Fitness Value}\ (X_i) - \text{Fitness Value}\ (X_{i+1}) \leq 0.0001. \quad (25)$$

4. Computational Experiments

4.1. Model Development. In this section, the results of multiobjective fuzzy mixed assembly line sequencing model are presented based on the development of new test-beds. According to Silberholz and Golden [49], new test-beds are developed when an existing test-bed is insufficient. There are two points which need to be addressed when developing new test-beds, that is, the purpose of developing the test-beds as well as the accessibility of new test instances [49]. The purpose of a problem suite is to emulate real-world problem instances with a variety of test cases and difficulty levels. When creating a new test-bed, the focus is to provide others with access to problem instances. This enables other researchers to make comparisons easily, while ensuring that the problem instances are widely used. One way to ensure this is to create a simple generating function for the problem instances. Capturing the real aspects of a problem is particularly significant when developing a new test-bed.

4.1.1. Input Data. A hypothetical numerical test-bed example is designed to test the fuzzy mixed-model assembly line sequencing problem. The input data of the hypothetical numerical example are given in Table 7, consisting of 50 jobs and 20 parts in order to produce four products. There are five machine tools (one lathe, two CNC, and two robots) assigned to assemble four products.

4.1.2. Initialization of Parameters and Fuzzy Variables. The initialization of parameters for the mixed-model assembly line sequencing example is shown in Table 8. It can be seen that 50 jobs (J_i, $i = 1, 2, \ldots, 50$) are required to produce 20 parts (P_j, $j = 0, 1, \ldots, 19$) and these parts are assembled to produce four types of products (P.No_k, $k = 1, 2, 3, 4$). The number of tool changes is 5 (labelled as A, B, C, D, and E) and these tools are assigned to five machines ($M(X_i)$, $i = 0, 1, 2, 3, 4$). The job sequence is dependent upon the part and product assembly and is described as follows. First, the job number is assigned to produce the first product, ranging from 1 to 10 (P.No_1, $J_i = 1, 2, \ldots, 10$). It shall be highlighted that there are 10 jobs in this case and they are sequenced to produce three parts according to the following order. Job numbers 1, 2, and 3 are assigned to produce Part_0 ((P_i), $i = 0 \rightarrow J_1, J_2, J_3$), while job numbers 4 and 5 are assigned to produce Part_1 ((P_i), $i = 1 \rightarrow J_4, J_5$). Job numbers 6 and 7 are assigned to produce Part_2 ((P_i), $i = 2 \rightarrow J_6, J_7$), whereas job numbers 8, 9, and 10 are assigned to produce Part_3, which is a subpart of the product assembly ((P_i), $i = 3 \rightarrow J_8, J_9, J_{10}$). Production of the second, third, and fourth products is based on the sequence described for the first product, as shown in Table 8. The fuzzy processing time of each job (J_i, $i = 1, 2, \ldots, 10$) is defined as a triplet (a_1, a_2, a_3). The total operating time is based on the fuzzy triangular time, and is required to complete the jobs sequentially when producing each part, in which each part is assigned to a machine (M_i, $i = 0, 1, 2, 3, 4$). The total operating time is defined as $(\widetilde{\text{OP} \cdot P_i}(a_1, a_2, a_3)$, $i = 0, 1, \ldots, 19$; $a = \text{time})$. The total travelling time based on the above information is defined as $(\widetilde{\text{T.t} \cdot P_j}(a_1, a_2, a_3)$, $j = 0, 1, \ldots, 19$; $a = \text{time})$. The operation and travelling time is fuzzy numbers, which are indicated by a_1, a_2, and a_3. The parameters a_1, a_2, and a_3 represent the optimistic time, normal time, and pessimistic time, respectively.

The development of a model for mixed-model assembly line sequencing is presented in Figure 6, based on the parameters listed in Table 8. The solid arrows ($\rightarrow$) represent the order of the product line (sequence of part production), discrete part manufacturing assembly, leading to the finished products, as indicated by the job numbers. Suppose that the production process involves manufacturing four products using the same assembly line. In other words, four different products are manufactured simultaneously on the assembly line, and hence the problem is a mixed-model assembly line problem. In this example, 20 parts need to be manufactured using five machines. From Figure 6, the order of the production of parts is represented by the dashed rectangles and is termed as the process line. The assembly lines are represented by the dotted rectangles. The assignment of parts to their respective machines based on job number is illustrated in Figure 7.

4.1.3. Multiobjective Evaluation. The final results based on the existing and optimized data are shown in Table 9. The overall results show that the existing fuzzy data is improved by optimization. The total fuzzy existing scheduling time is optimized from (166, 250, 266) to (62.5, 73, 88.5). The total fuzzy setup numbers for the existing and optimized data are (44, 44, 44) and (37, 37, 39), respectively. The total fuzzy existing efficient frontier is (105, 147, 155), while the optimized one is (49.25, 54.5, 63.75). The total fuzzy existing scheduling and setup time are optimized from (254, 338, 354) to (135, 145, 168). The total fuzzy operation setup time for the existing and optimized data are (88, 88, 88) and (72, 72, 78), respectively. The total fuzzy existing changing setup cost is reduced from (\$3520, \$3520, \$3520) to (\$2880, \$2880, \$3120). The total fuzzy unit produced per day for the existing and optimized data is approximately (1.80, 1.92, 2.89) and (5.33, 6.58, 7.62), respectively. The total existing fuzzy percentage efficiency is optimized from (10.32%, 10.90%, 11.45%) to (30.16%, 32.22%, 35.34%).

4.2. Method Development. The overall results obtained from the dynamic Fuzzy GA method are presented in this section, based on the hypothetical numerical example (test-beds) described in Section 4.1. The optimum fuzzy population size and generation size, as well as the selection of a suitable tournament candidate, crossover rate, and mutation rate, are discussed. The results obtained from both GA and fuzzy GA methods are also compared and discussed in detail in this section. The fitness values generated based on the fuzzy rule base with respect to the number of generations and population size are shown in Figure 8. The purpose of the comparative analysis is to select the best population size and number of generations, based on the job number (chromosome size) of the mixed-model assembly line sequencing problem. Figure 8 shows the fitness values for a chromosome size of 50, which indicates that 50 jobs are required to produce four products with a 19-part assembly assigned to five machines. It is found that the fitness value is 58 when the population size is 10 and the number of generations is 539. The fitness value increases slightly with a value of 59 when the population size and the number of generations is 40 and 652, respectively. The fitness values are determined to be 58 and 57 for a population size of 80 and 160, and the corresponding number of generations is 61 and 416, respectively. The results show that for a chromosome size of 50, the highest fitness value is found to be 54.5 when the population size is 100 and the number of generations is 83.

The results of the tournament candidate, crossover rate, and mutation rate for a mixed-model line sequencing problem with 50 chromosomes (50 jobs), fixed population size (of 100), and 800 generations are presented in Figure 9. The results are compared with the dynamic fuzzy tournament candidate and crossover and mutation rates. A comparison between the performance of the fixed tournament candidate (mutation rate: 0.02 and crossover rate: 0.5) and dynamic

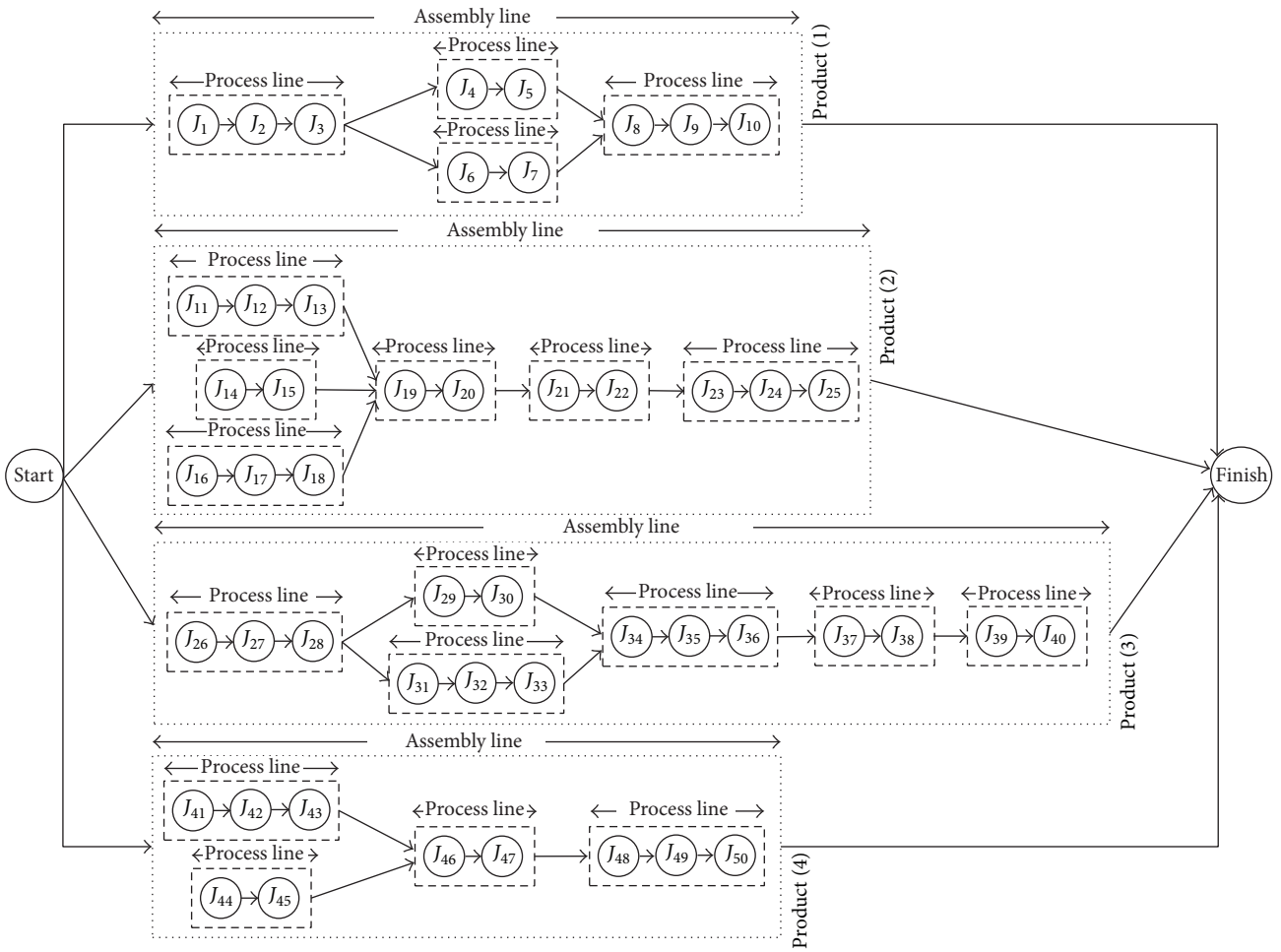


Figure 6: Example of mixed-model assembly line sequencing (50 jobs, 20 parts, and 4 products).

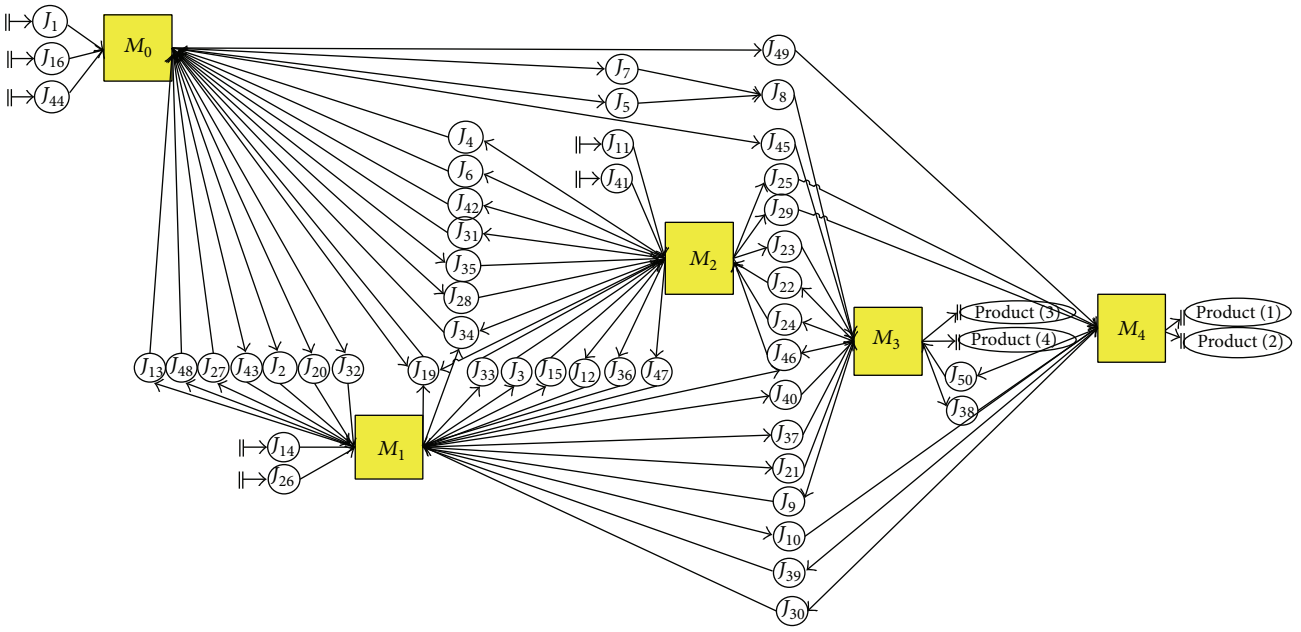


Figure 7: Assignment of parts to their respective machines based on the example for mixed-model assembly line sequencing.

Table 7: Input data for numerical example.

Number of jobs	Number of products	Number of parts	Number of lathe machines, CNC, and robots	Number of machine tools
50	4	20	5	5

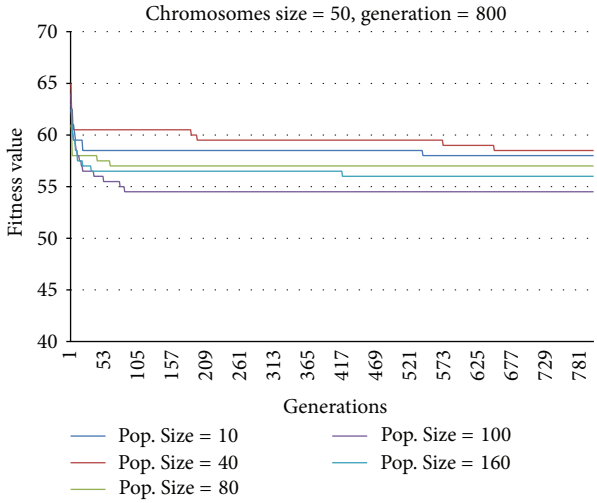


Figure 8: Fitness values versus number of generations for a chromosome size of 50.

fuzzy tournament candidate (in which the tournament candidate is varied from 2 to 5) is shown in Figure 9(a). The results show that the selection of tournament candidates affects the performance of the GA. It can be observed that a decrease in the number of tournament candidates improves the convergence of the GA at the expense of reduced accuracy. A comparison between the performance of the GA having a fixed crossover rate (tournament candidate: 3 and mutation rate: 0.02) and dynamic fuzzy crossover rate (whereby the crossover rate is changed from 0.25 to 1) is shown in Figure 9(b). Similarly, it can be seen that changes in the crossover rate have a significant influence on the results, whereby a high crossover rate yields faster convergence with reduced accuracy. From Figure 9(b), a crossover rate of 1 yields faster convergence compared to a crossover rate of 0.9. However, a crossover rate of 0.25 offers higher accuracy. A comparison between the performance of the GA with fixed mutation rate (tournament candidate: 3 and crossover rate: 0.5) and dynamic fuzzy mutation rate (in which the mutation rate is varied from 0.0 to 0.04) is shown in Figure 9(c). It is also evident that the mutation rate has a significant influence on the performance of the GA. It can be noted that a lower mutation rate yields faster convergence at the expense of reduced accuracy. From Figure 9(c), it can be seen that a mutation rate of 0.01 gives faster convergence compared to a mutation rate of 0.04; however, the latter mutation rate offers higher accuracy. The most significant finding of this study is that the dynamic fuzzy tournament candidate, crossover rate, and mutation rate yield faster convergence with higher accuracy in the optimum fitness values.

The optimum fitness values with respect to the number of generations and time taken to achieve convergence for various tournament candidates, crossover rates, and mutation rates are summarized in Table 10. From Table 10, the fastest convergence is achieved when the number of generations is 60, whereas the optimum fitness value is more accurate when the number of generations is 56 for the dynamic fuzzy tournament candidate. The lowest fuzzy GA execution time is 1.45 s. It can be observed from Table 10 that faster convergence is attained for the dynamic fuzzy crossover and mutation rates when the number of generations is 79 and 147, respectively. However, the optimum fitness values for both crossover and mutation rates have a higher accuracy when the number of generations is 55. The lowest fuzzy GA execution time is found to be 2.50 and 3.82 s, respectively.

A comparison of the static and dynamic behaviors of the tournament candidate, crossover rate, and mutation rate between conventional GA and FGA is shown in Figure 10. The conventional GA is used for solving the mixed-model assembly line sequencing problem and comprises the following parameters (chromosome size: 50, population size: 100, number of generations: 800, tournament candidate: 3, crossover rate: 0.5, and mutation rate: 0.02). The dynamic FGA is also implemented for solving the above problem using the same chromosome size, population size, and number of generations as for the conventional GA. However, it shall be emphasized that fuzzy tournament candidate, fuzzy crossover rate, and fuzzy mutation rate are used as the parameters for dynamic FGA. The results show that the tournament candidate, crossover rate, and mutation rate are dynamically and automatically modified during the optimization process using the fuzzy logic controller. Figure 11 and Table 11 show the final results of the dynamic FGA and conventional GA based on the optimum fitness value and number of generations. The FGA designed with three fuzzy dynamic parameter controllers (i.e., tournament candidate, crossover rate, and mutation rate) exhibits a superior performance compared to the existing GA. The results reveal that the FGA is capable of rapid and efficient searching compared to the standard GA for solving the mixed-model assembly line sequencing problem. It is evident from the results that the optimum fitness value for dynamic FGA (54.5) is more accurate than the conventional GA (57.5). The FGA yields faster convergence, whereby the number of generations is 87. In contrast, convergence is achieved only when the number of generations is 243 for conventional GA. The FGA also gives a lower execution time of 2.15 s compared to the GA, which has an execution time of 6.31 s.

Table 8: Fuzzy variables and initialization of parameters.

X_i	J_i	P.NO	A.S	S.No	$P(X_i)$	$M(X_i)$	OP			T.t		
1	J_1	$P.NO_1$	AS_0	A	P_0	M_0	1.50	2	2.50	0.50	1	1.5
2	J_2	$P.NO_1$	AS_0	A	P_0	M_1	2.50	3	3.50	4.50	5	5.5
3	J_3	$P.NO_1$	AS_0	A	P_0	M_2	0.50	1	1.50	0.50	1	1.5
4	J_4	$P.NO_1$	AS_1	B	P_1	M_0	0.50	1	1.50	1.50	2	2.5
5	J_5	$P.NO_1$	AS_1	B	P_1	M_2	1.50	2	2.50	2.50	3	3.5
6	J_6	$P.NO_1$	AS_1	A	P_2	M_0	1.50	2	2.50	2.50	3	3.5
7	J_7	$P.NO_1$	AS_1	A	P_2	M_2	0.50	1	1.50	2.50	3	3.5
8	J_8	$P.NO_1$	AS_2	C	P_3	M_3	1.50	2	2.50	3.50	4	4.5
9	J_9	$P.NO_1$	AS_2	A	P_3	M_1	2.50	3	3.50	4.50	5	5.5
10	J_{10}	$P.NO_1$	AS_2	A	P_3	M_4	1.50	2	2.50	0.50	1	1.5
11	J_{11}	$P.NO_2$	AS_0	E	P_4	M_2	3.50	4	4.50	1.50	2	2.5
12	J_{12}	$P.NO_2$	AS_0	C	P_4	M_1	2.50	3	3.50	0.50	1	1.5
13	J_{13}	$P.NO_2$	AS_0	E	P_4	M_0	1.50	2	2.50	2.50	3	3.5
14	J_{14}	$P.NO_2$	AS_0	E	P_5	M_1	0.50	1	1.50	1.50	2	2.5
15	J_{15}	$P.NO_2$	AS_0	B	P_5	M_2	0.50	1	1.50	0.50	1	1.5
16	J_{16}	$P.NO_2$	AS_0	B	P_6	M_0	1.50	2	2.50	3.50	4	4.5
17	J_{17}	$P.NO_2$	AS_0	B	P_6	M_3	1.50	2	2.50	0.50	1	1.5
18	J_{18}	$P.NO_2$	AS_0	B	P_6	M_1	2.50	3	3.50	1.50	2	2.5
19	J_{19}	$P.NO_2$	AS_1	C	P_7	M_0	2.50	3	3.50	2.50	3	3.5
20	J_{20}	$P.NO_2$	AS_1	C	P_7	M_1	0.50	1	1.50	1.50	2	2.5
21	J_{21}	$P.NO_2$	AS_2	A	P_8	M_3	0.50	1	1.50	0.50	1	1.5
22	J_{22}	$P.NO_2$	AS_2	A	P_8	M_2	4.50	5	5.50	1.50	2	2.5
23	J_{23}	$P.NO_2$	AS_3	D	P_9	M_3	2.50	3	3.50	2.50	3	3.5
24	J_{24}	$P.NO_2$	AS_3	B	P_9	M_2	1.50	2	2.50	1.50	2	2.5
25	J_{25}	$P.NO_2$	AS_3	D	P_9	M_4	0.50	1	1.50	4.50	5	5.5
26	J_{26}	$P.NO_3$	AS_0	B	P_{10}	M_1	3.50	4	4.50	0.50	1	1.5
27	J_{27}	$P.NO_3$	AS_0	A	P_{10}	M_0	2.50	3	3.50	0.50	1	1.5
28	J_{28}	$P.NO_3$	AS_0	C	P_{10}	M_2	1.50	2	2.50	2.50	3	3.5
29	J_{29}	$P.NO_3$	AS_1	A	P_{11}	M_4	0.50	1	1.50	1.50	2	2.5
30	J_{30}	$P.NO_3$	AS_1	A	P_{11}	M_1	1.50	2	2.50	1.50	2	2.5
31	J_{31}	$P.NO_3$	AS_1	C	P_{12}	M_0	0.50	1	1.50	1.50	2	2.5
32	J_{32}	$P.NO_3$	AS_1	C	P_{12}	M_1	1.50	2	2.50	0.50	1	1.5
33	J_{33}	$P.NO_3$	AS_1	C	P_{12}	M_2	2.50	3	3.50	2.50	3	3.5
34	J_{34}	$P.NO_3$	AS_2	B	P_{13}	M_0	3.50	4	4.50	3.50	4	4.5
35	J_{35}	$P.NO_3$	AS_2	E	P_{13}	M_2	4.50	5	5.50	4.50	5	5.5
36	J_{36}	$P.NO_3$	AS_2	E	P_{13}	M_1	1.50	2	2.50	0.50	1	1.5
37	J_{37}	$P.NO_3$	AS_3	E	P_{14}	M_3	0.50	1	1.50	4.50	5	5.5
38	J_{38}	$P.NO_3$	AS_3	B	P_{14}	M_4	2.50	3	3.50	2.50	3	3.5
39	J_{39}	$P.NO_3$	AS_4	B	P_{15}	M_1	4.50	5	5.50	1.50	2	2.5
40	J_{40}	$P.NO_3$	AS_4	B	P_{15}	M_3	3.50	4	4.50	0.50	1	1.5
41	J_{41}	$P.NO_4$	AS_0	A	P_{16}	M_2	3.50	4	4.50	0.50	1	1.5
42	J_{42}	$P.NO_4$	AS_0	D	P_{16}	M_0	2.50	3	3.50	2.50	3	3.5
43	J_{43}	$P.NO_4$	AS_0	A	P_{16}	M_1	1.50	2	2.50	1.50	2	2.5
44	J_{44}	$P.NO_4$	AS_0	B	P_{17}	M_0	0.50	1	1.50	3.50	4	4.5
45	J_{45}	$P.NO_4$	AS_0	B	P_{17}	M_3	0.50	1	1.50	0.50	1	1.5
46	J_{46}	$P.NO_4$	AS_1	A	P_{18}	M_2	3.50	4	4.50	1.50	2	2.5
47	J_{47}	$P.NO_4$	AS_1	B	P_{18}	M_1	2.50	3	3.50	1.50	2	2.5
48	J_{48}	$P.NO_4$	AS_2	B	P_{19}	M_0	1.50	2	2.50	2.50	3	3.5
49	J_{49}	$P.NO_4$	AS_2	A	P_{19}	M_4	0.50	1	1.50	1.50	2	2.5
50	J_{50}	$P.NO_4$	AS_2	B	P_{19}	M_3	3.50	4	4.50	2.50	3	3.5

Table 9: Summary of the final results based on existing and optimized data.

	Existing data (Opt., Med., Pes.)	Optimized data (Opt., Med., Pes.)
Total scheduling time	(166, 250, 266)	(62.5, 73, 88.5)
Total setup number (No.)	(44, 44, 44)	(37, 37, 39)
Total efficient frontiers	(105, 147, 155)	(49.25, 54.5, 63.75)
Total scheduling time with setup time	(254, 338, 354)	(135, 145, 168)
Total operation setup time	(88, 88, 88)	(72, 72, 78)
Total changing setup cost ($)	(3520, 3520, 3520)	(2880, 2880, 3120)
Total units produced per day	(2.89, 1.92, 1.80)	(7.62, 6.58, 5.33)
Total efficiency (%)	(11.45%, 10.32%, 10.90%)	(30.16%, 35.34%, 32.22%)
Total idle time (%)	(88.55%, 89.68%, 89.10%)	(69.84%, 64.66%, 67.78%)

Table 10: Comparison of optimum fitness values with respect to the number of generations and time taken to achieve convergence for various tournament candidates and crossover and mutation rates.

	Tournament candidates					
	Tournament candidates (2)	Tournament candidates (3)	Tournament candidates (4)	Tournament candidates (5)	Tournament candidates (*fuzzy)	
Value	57	57	56.5	56.5	56	
Generation	145	96	272	73	60	
Time	1.48	1.48	1.46	1.46	1.45	
	Crossover rate					
	Crossover rate (0.25)	Crossover rate (0.5)	Crossover rate (0.7)	Crossover rate (0.8)	Crossover rate (0.9)	Crossover rate (1)	Crossover rate (*fuzzy)
Value	56	58.5	56.5	56.5	56.5	56.5	55
Generation	269	45	392	127	80	64	79
Time	6.99	1.17	10.19	3.30	2.08	1.66	2.50
	Mutation rate						
	Mutation rate (0.0)	Mutation rate (0.01)	Mutation rate (0.02)	Mutation rate (0.03)	Mutation rate (0.04)	Mutation rate (0.05)	Mutation rate (*fuzzy)
Value	61	57.5	56.5	56	55.5	55.5	55
Generation	48	188	154	527	529	479	147
Time	1.25	4.88	4.01	13.70	13.75	12.45	3.82

Table 11: Comparison of GA versus FGA with respect to fitness value, number of generations, and time taken to achieve convergence for various control parameters.

	GA	FGA
Chromosome size	50	FES
Population size	100	FES
Tournament candidate	3	ALDFC
Crossover	0.5	ALDFC
Mutation	0.02	ALDFC
Generation	243	83
Value	57.5	54.5
Time	6.31″	2.15″

5. Conclusion

It is known that mixed-model assembly line sequencing is a problem with multiple conflicting objectives. A mixed-model assembly line sequencing optimization model is developed in order to address two conflicting objectives, namely, minimizing the make-span (i.e., minimizing scheduling time, travelling time, and machine idle time and maximizing production) and minimizing the setup time (i.e., minimizing the number of machine setup tool change and minimizing the machine setup cost) simultaneously, which occur when switching between different products. These objectives have been achieved successfully and tested using a hypothetical numerical test-bed. The hypothetical numerical test-bed involves 50 jobs to produce 20 parts using five machines in order to assemble four products. Triangular and trapezoidal fuzzy numbers are applied to the operation time and travelling time variables. The fuzzy numbers are categorized as optimistic, medium, and pessimistic fuzzy total scheduling time. The results show that the fuzzy total scheduling time (166, 250, and 266) decreases to (62.5, 73, and 88.5) after optimization. Comparison is made between the existing and optimized results representing the efficiency and idle time of

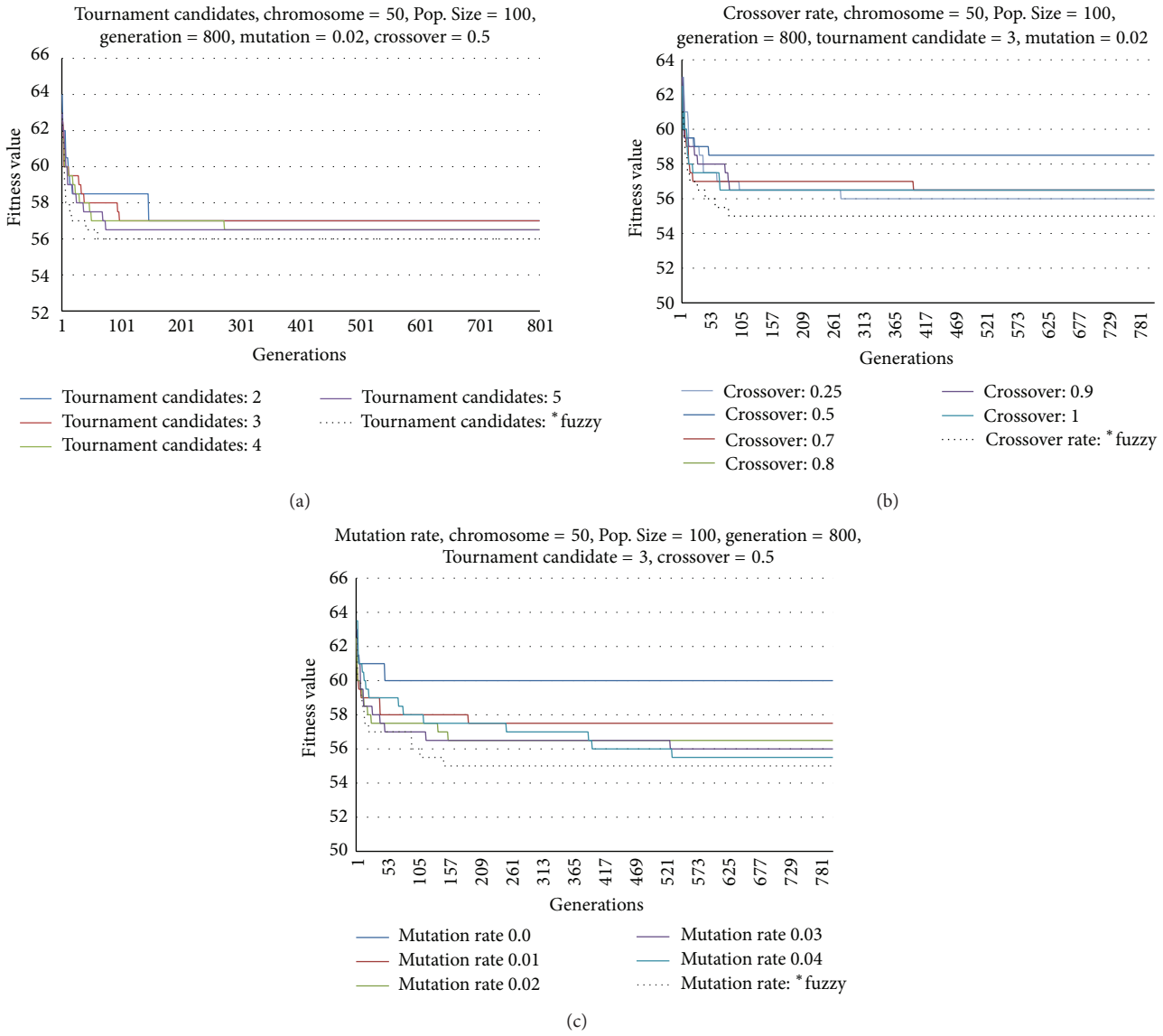


Figure 9: Comparison between tournament candidates and crossover and mutation rates.

each machine. The existing and optimized results of the total scheduling time, total setup number, total efficient frontier, total scheduling time with setup time, total operation setup time, total changing setup cost ($), and total number of units produced per day are also compared.

In this study, a dynamic fuzzy GA approach is proposed, in which a fuzzy expert experience controller (FEEC) and automatic learning dynamic fuzzy controller (ALDFC) are integrated with genetic algorithm in order to solve a multiobjective mixed-model assembly line sequencing problem. The aim of developing this method is to enhance the performance and effectiveness of GA. The fuzzy expert experience controller is used in order to decide the number of generations and population size in the GA. In order to dynamically control the genetic operators, three automatic learning dynamic fuzzy controllers are implemented to select the number of tournament candidates and set the crossover and mutation rates based on existing conditions. The performance of the FGA has been evaluated with respect to the adaptive control parameters. The results show that the FGA exhibits superior performance compared to the conventional GA. The FGA is capable of searching faster and more efficiently compared to the standard GA when solving the mixed-model assembly line sequencing problem due to the following reasons.

(1) The population size and control parameters can be chosen appropriately for the problem under investigation.

(2) The control parameters can be adjusted on-line to adapt dynamically to new situations.

(3) The FGA can assist users in accessing, designing, implementing, and validating genetic algorithms for a given task.

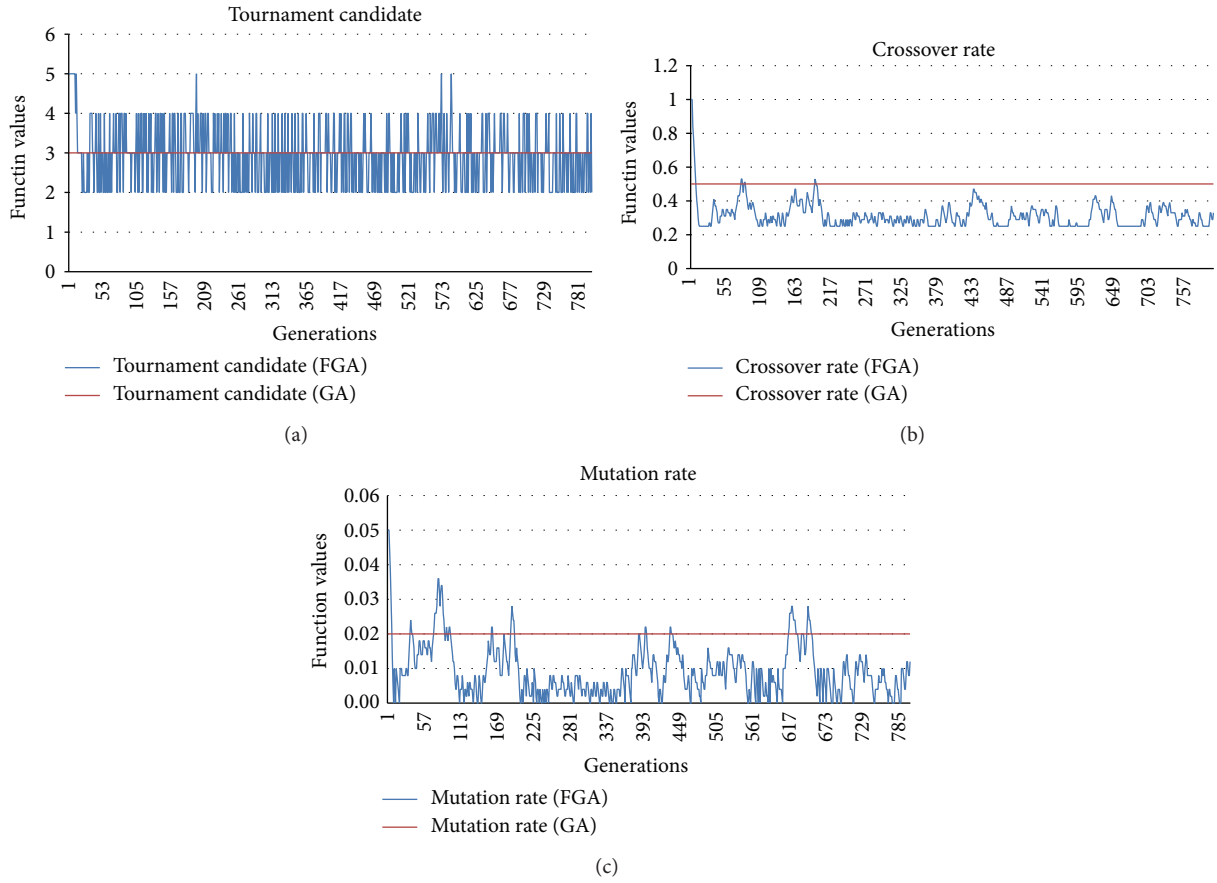


Figure 10: Comparison of behavior between fixed and dynamic fuzzy tournament candidates and crossover and mutation rates.

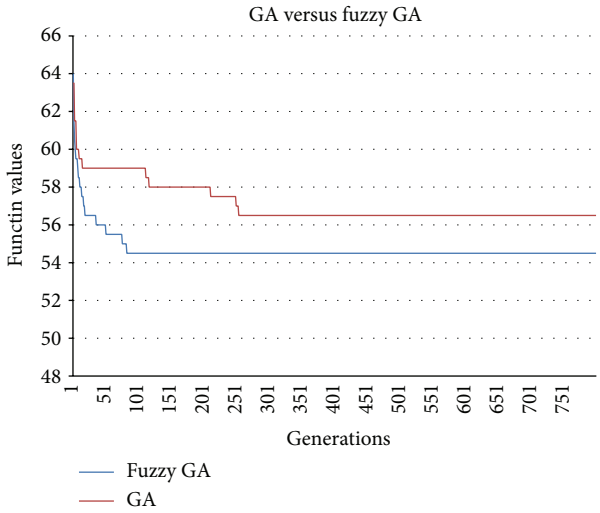


Figure 11: Final results of GA versus FGA.

The optimal fitness value for the dynamic FGA (54.5) is more accurate compared to that for conventional GA (57.5). The FGA attains faster convergence (number of generations: 87) whereas the conventional GA achieves convergence when the number of generations is 243. The FGA also yields a lower execution time (2.15 s) compared to the conventional GA (6.31 s).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to acknowledge the Malaysian Ministry of Higher Education (MOHE) for their financial support under High Impact Research Grant (no. UM.C/HIR/MOHE/ENG/35 (D000035-16001)).

References

[1] G. Heike, M. Ramulu, E. Sorenson, P. Shanahan, and K. Moinzadeh, "Mixed model assembly alternatives for low-volume manufacturing: the case of the aerospace industry," *International Journal of Production Economics*, vol. 72, no. 2, pp. 103–120, 2001.

[2] A. R. Rahimi-Vahed, M. Rabbani, R. Tavakkoli-Moghaddam, S. A. Torabi, and F. Jolai, "A multi-objective scatter search for a mixed-model assembly line sequencing problem," *Advanced Engineering Informatics*, vol. 21, no. 1, pp. 85–99, 2007.

[3] B. Javadi, A. Rahimi-Vahed, M. Rabbani, and M. Dangchi, "Solving a multi-objective mixed-model assembly line sequencing problem by a fuzzy goal programming approach," *The International Journal of Advanced Manufacturing Technology*, vol. 39, no. 9-10, pp. 975–982, 2008.

[4] N. Boysen, M. Fliedner, and A. Scholl, "Sequencing mixed-model assembly lines: survey, classification and model critique," *European Journal of Operational Research*, vol. 192, no. 2, pp. 349–373, 2009.

[5] J. F. Bard, E. Dar-El, and A. Shtub, "Analytic framework for sequencing mixed model assembly lines," *The International Journal of Production Research*, vol. 30, no. 1, pp. 35–48, 1992.

[6] Z. Taha, F. Tahriri, and A. Zuhdi, "Job sequencing and layout optimization in virtual production line," *Journal of Quality*, vol. 18, no. 4, pp. 351–374, 2011.

[7] E. M. Dar-El, "Mixed-model assembly line sequencing problems," *Omega*, vol. 6, no. 4, pp. 313–323, 1978.

[8] E. M. Dar-El and R. F. Cother, "Assembly line sequencing for model mix," *The International Journal of Production Research*, vol. 13, no. 5, pp. 463–477, 1975.

[9] E. M. Dar-El and A. Nadivi, "A mixed-model sequencing application," *The The International Journal of Production Research*, vol. 19, no. 1, pp. 69–84, 1981.

[10] C. Merengo, F. Nava, and A. Pozzetti, "Balancing and sequencing manual mixed-model assembly lines," *The International Journal of Production Research*, vol. 37, no. 12, pp. 2835–2860, 1999.

[11] X. Zhu, S. J. Hu, Y. Koren, S. P. Marin, and N. Huang, "Sequence planning to minimize complexity in mixed-model assembly lines," in *Proceedings of the IEEE International Symposium on Assembly and Manufacturing (ISAM '07)*, pp. 251–258, July 2007.

[12] B. Rekiek, P. de Lit, and A. Delchambre, "Designing mixed-product assembly lines," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 3, pp. 268–280, 2000.

[13] P. de Lit, A. Delchambre, and J.-M. Henrioud, "An integrated approach for product family and assembly system design," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 324–334, 2003.

[14] J. Bukchin, E. M. Dar-El, and J. Rubinovitz, "Mixed model assembly line design in a make-to-order environment," *Computers & Industrial Engineering*, vol. 41, no. 4, pp. 405–421, 2002.

[15] A. Khan and A. J. Day, "A knowledge based design methodology for manufacturing assembly lines," *Computers & Industrial Engineering*, vol. 41, no. 4, pp. 441–467, 2001.

[16] A. R. Mendes, A. L. Ramos, A. S. Simaria, and P. M. Vilarinho, "Combining heuristic procedures and simulation models for balancing a PC camera assembly line," *Computers & Industrial Engineering*, vol. 49, no. 3, pp. 413–431, 2005.

[17] A. Joly and Y. Frein, "Heuristics for an industrial car sequencing problem considering paint and assembly shop objectives," *Computers & Industrial Engineering*, vol. 55, no. 2, pp. 295–310, 2008.

[18] A. Norozi, M. K. Ariffin, and N. Ismail, "Application of intelligence based genetic algorithm for job sequencing problem on parallel mixed-model assembly line," *The American Journal of Engineering and Applied Sciences*, vol. 3, no. 1, pp. 831–840, 2010.

[19] S. Ghosh and R. J. Gagnon, "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems," *The International Journal of Production Research*, vol. 27, no. 4, pp. 637–670, 1989.

[20] W. K. Wong, C. K. Chan, and W. H. Ip, "Optimization of spreading and cutting sequencing model in garment manufacturing," *Computers in Industry*, vol. 43, no. 1, pp. 1–10, 2000.

[21] C. Moon, M. Lee, Y. Seo, and Y. H. Lee, "Integrated machine tool selection and operation sequencing with capacity and precedence constraints using genetic algorithm," *Computers & Industrial Engineering*, vol. 43, no. 3, pp. 605–621, 2002.

[22] B. S. P. Reddy and C. S. P. Rao, "A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS," *The International Journal of Advanced Manufacturing Technology*, vol. 31, no. 5-6, pp. 602–613, 2006.

[23] M. Gen, R. Chen, and L. Lin, *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*, Springer, 2008.

[24] S. G. Ponnambalam, P. Aravindan, and M. Subba Rao, "Genetic algorithms for sequencing problems in mixed model assembly lines," *Computers & Industrial Engineering*, vol. 45, no. 4, pp. 669–690, 2003.

[25] P. T. Zacharia and A. C. Nearchou, "Multi-objective fuzzy assembly line balancing using genetic algorithms," *Journal of Intelligent Manufacturing*, vol. 23, no. 3, pp. 615–627, 2012.

[26] P. R. McMullen, "A Kohonen self-organizing map approach to addressing a multiple objective, mixed-model JIT sequencing problem," *International Journal of Production Economics*, vol. 72, no. 1, pp. 59–71, 2001.

[27] P. R. McMullen, "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives," *Artificial Intelligence in Engineering*, vol. 15, no. 3, pp. 309–317, 2001.

[28] P. R. McMullen, "An efficient frontier approach to addressing JIT sequencing problems with setups via search heuristics," *Computers & Industrial Engineering*, vol. 41, no. 3, pp. 335–353, 2001.

[29] P. R. McMullen and G. V. Frazier, "A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line," *IIE Transactions*, vol. 32, no. 8, pp. 679–686, 2000.

[30] V. Giard and J. Jeunet, "Optimal sequencing of mixed models with sequence-dependent setups and utility workers on an assembly line," *International Journal of Production Economics*, vol. 123, no. 2, pp. 290–300, 2010.

[31] S.-M. Im and J.-J. Lee, "Adaptive crossover, mutation and selection using fuzzy system for genetic algorithms," *Artificial Life and Robotics*, vol. 13, no. 1, pp. 129–133, 2008.

[32] R. Subbu, A. C. Sanderson, and P. P. Bonissone, "Fuzzy logic controlled genetic algorithms versus tuned genetic algorithms: an agile manufacturing application," in *Proceedings of the IEEE International Symposium on Intelligent Control (ISIC '98)*, pp. 434–440, September 1998.

[33] Y. Yun and M. Gen, "Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics," *Fuzzy Optimization and Decision Making*, vol. 2, no. 2, pp. 161–175, 2003.

[34] F. Herrera and M. Lozano, "Adaptation of genetic algorithm parameters based on fuzzy logic controllers," in *Genetic Algorithms and Soft Computing*, pp. 95–129, 1996.

[35] F. Herrera and M. Lozano, "Adaptive genetic algorithms based on fuzzy techniques," in *Genetic Algorithms and Soft Computing*, pp. 775–780, 1996.

[36] H. Y. Xu and G. Vukovich, "Fuzzy genetic algorithm with effective search and optimization," in *Proceedings of International Joint Conference on Neural Networks*, pp. 2967–2970, October 1993.

[37] Y. Song, G. Wang, P. Wang, and A. Johns, "Environmental/econoimic dispatch using fuzzy logic controlled genetic algorithms," *IEEE Proceedings on Generation, Transmission and Distribution*, vol. 144, no. 4, pp. 377–382, 1997.

[38] K. L. Mak, Y. S. Wong, and X. X. Wang, "Adaptive genetic algorithm for manufacturing cell formation," *The International Journal of Advanced Manufacturing Technology*, vol. 16, no. 7, pp. 491–497, 2000.

[39] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[40] Q. H. Wu, Y. J. Cao, and J. Y. Wen, "Optimal reactive power dispatch using an adaptive genetic algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 20, no. 8, pp. 563–569, 1998.

[41] P. T. Wang, G. S. Wang, and Z. G. Hu, "Speeding up the search process of genetic algorithm by fuzzy logic," in *Proccedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, pp. 665–671, 1997.

[42] Z. X. Guo, W. K. Wong, S. Y. S. Leung, J. T. Fan, and S. F. Chan, "Mathematical model and genetic optimization for the job shop scheduling problem in a mixed- and multi-product assembly environment: a case study based on the apparel industry," *Computers & Industrial Engineering*, vol. 50, no. 3, pp. 202–219, 2006.

[43] W. Cheung and H. Zhou, "Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times," *Annals of Operations Research*, vol. 107, no. 1–4, pp. 65–81, 2001.

[44] G. Mosheiov, A. Sarig, and J. Sidney, "The Browne-Yechiali single-machine sequence is optimal for flow-shops," *Computers & Operations Research*, vol. 37, no. 11, pp. 1965–1967, 2010.

[45] S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, and W. Kubiak, "Sequencing of parts and robot moves in a robotic cell," *International Journal of Flexible Manufacturing Systems*, vol. 4, no. 3-4, pp. 331–358, 1992.

[46] M. Gen, Y. Tsujimura, and Y. Li, "Fuzzy assembly line balancing using genetic algorithms," *Computers & Industrial Engineering*, vol. 31, no. 3-4, pp. 631–634, 1996.

[47] D. J. Fonseca, C. L. Guest, M. Elam, and C. L. Karr, "A fuzzy logic approach to assembly line balancing," *Mathware & Soft Computing*, vol. 12, no. 1, pp. 57–74, 2005.

[48] E. H. Mamdani and S. Assilian, "Experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[49] J. Silberholz and B. Golden, "Comparison of metaheuristics," in *Handbook of Metaheuristics*, Springer, 2010.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 756178, 22 pages
http://dx.doi.org/10.1155/2014/756178

Research Article

A Time Scheduling Model of Logistics Service Supply Chain Based on the Customer Order Decoupling Point: A Perspective from the Constant Service Operation Time

Weihua Liu, Yi Yang, Haitao Xu, Xiaoyan Liu, Yijia Wang, and Zhicheng Liang

College of Management and Economics, Tianjin University, Tianjin 300072, China

Correspondence should be addressed to Weihua Liu; lwhliu888@163.com

Received 31 August 2013; Accepted 30 December 2013; Published 12 March 2014

Academic Editors: J. G. Barbosa and M. D. Toksari

Copyright © 2014 Weihua Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In mass customization logistics service, reasonable scheduling of the logistics service supply chain (LSSC), especially time scheduling, is benefit to increase its competitiveness. Therefore, the effect of a customer order decoupling point (CODP) on the time scheduling performance should be considered. To minimize the total order operation cost of the LSSC, minimize the difference between the expected and actual time of completing the service orders, and maximize the satisfaction of functional logistics service providers, this study establishes an LSSC time scheduling model based on the CODP. Matlab 7.8 software is used in the numerical analysis for a specific example. Results show that the order completion time of the LSSC can be delayed or be ahead of schedule but cannot be infinitely advanced or infinitely delayed. Obtaining the optimal comprehensive performance can be effective if the expected order completion time is appropriately delayed. The increase in supply chain comprehensive performance caused by the increase in the relationship coefficient of logistics service integrator (LSI) is limited. The relative concern degree of LSI on cost and service delivery punctuality leads to not only changes in CODP but also to those in the scheduling performance of the LSSC.

1. Introduction

At present, with the growing demand for customized logistics services, many logistics enterprises not only provide customers with mass service but also meet the demand for customized service as well as consider changes in the logistics service mode. Specially, these enterprises attempt to provide mass customization logistics services (MCLS) instead of mass logistics services [1]. In the MCLS environment, to meet the individualized service requirements of customers and achieve the necessary capabilities for offering mass service, many logistics enterprises form a logistics service supply chain (LSSC) by means of unions and integration [2, 3]. In MCLS, the key factor in realizing the competitiveness of the LSSC is whether it can offer customized service with the cost of mass service through reasonable scheduling.

The main participants of MCLS are customers, functional logistics service providers (FLSPs), and logistics service integrators (LSIs). The LSI, as the core enterprise of the LSSC, needs to handle the logistics services demand of multiple costumers, analyze different service processes required by each customer, and integrate these similar orders to achieve mass service effects. As different FLSPs have advantages both in different logistics service processes and different logistics service functions, LSIs need to integrate the different advantages of FLSPs to provide customers with customized logistics service. Thus, the effect of a customer order decoupling point (CODP) location on scheduling performance must be taken into consideration. Customer order decoupling point (CODP) is a concept frequently used in distinguishing Make-to-Stock (MTS) operations from Make-to-Order (MTO) operations [4]. The concept of CODP is widely used in the production and manufacturing fields and has become the main content of researches on postponement strategies.

The CODP is the demarcation point of mass service and customization service both in product supply chain field and service supply chain field, but significant differences exist in the CODPs of these two fields. On the one hand, in the product supply chain, CODP is always regarded as the inventory holding point of the component commonality

in order to meet customization demand from downstream customers quickly. In contrast, in the service supply chain there is no inventory as service cannot be stored. On the other hand, in the manufacturing supply chain, the component commonality before CODP usually could be produced ahead of time and held as work-in-process (WIP). On the contrary, in the service supply chain, service cannot be mass-produced ahead of schedule since service has a characteristic of inseparability [5]; that is, for service the production and consumption are usually going on at the same time. Thus, in service supply chain, either the process before the CODP or the one after the CODP depends on a specific customer order. Given the reasons above, the scheduling problem of the service supply chain confronts more dynamics and uncertainty when compared to the one of the manufacturing supply chain.

Completion time is an important index for logistics service level. The time scheduling problem is one of key problems in logistics service scheduling. Numerous logistics companies have paid much attention to improve the time scheduling performance while offering mass customization logistics service to customers. For example, the e-commerce transactions of China Taobao online mall amounted to 3.36 billion in November 11, 2011. In only one day, Yuantong Express Delivery received from all its branch companies in China up to 2.67 million parcels, which is more than four times the number in the same period in 2010. Express parcels must be delivered in a customized method within three to five days to customers, which are located in 31 provinces in China. This requirement caused tremendous pressure on Yuantong Express Delivery. Meeting the individual logistics service requirements within a certain period of time became a great problem for Yuantong Express Delivery. They needed to combine similar orders and set a CODP for all orders while considering the time requirement of costumers. Ahead of the CODP, Yuantong Express Delivery carried out mass service catering to all orders and then offered customized service to each order. Apparently, the influence of CODP location on the LSSC time scheduling performance should be considered. Another example is as follows: there are three customers in Tianjin city in request of delivery service to three cities in Eastern China, respectively, which are Hangzhou, Ningbo, and Jinhua city. The required delivery completion time is 48 hours for the order to Hangzhou, 60 hours for that of Ningbo, and 72 hours for that of Jinhua. Obviously, it is not economical to deliver these three orders, respectively, using different trucks. Thus, Yuantong Express Delivery always first consolidates the three orders and sends them to a place (such as Shanghai) from Tianjin in a time, which is mass delivery service. After arriving in Shanghai, customization service is conducted. Yuantong Express Delivery will assign different trucks to deliver the three orders' parcels, respectively, to Hangzhou, Ningbo, and Jinhua. Therefore, Shanghai becomes the CODP of the whole logistics process. In practice, the selectable CODP is not confined to Shanghai and other cities, such as Nanjing and Suzhou, may be alternative options. Various decisions of the CODP would lead to a variety of scheduling results. As this example indicates, the CODP is the demarcation point of mass service and customization service, and it directly determines the service process sequence, the service time, and the service cost. Thus, the impact of the CODP decision on the LSSC scheduling performance must be taken into consideration.

In our previous study [6], we have come up with the basic model of time scheduling in the LSSC and explored the delay coefficient of customer order completion time and relationship cost coefficient of the LSI on the time scheduling result in MC environment. In addition, we have compared the effects of two behaviors, decreasing the delay coefficient of customer order completion time and relationship cost coefficient of the LSI, which both can improve the supply chain performance to some extent. But what Liu et al. [6] considered was the operation situation after the CODP position has been decided which means that the CODP position is a known parameter and the CODP decision problem has not been taken into consideration. In actual scheduling operation, the time constraints will affect the CODP position option. Furthermore, the CODP position will have a great influence on the time scheduling performance. Thus, it is necessary to consider the influence of the CODP on time scheduling results. This paper delves into constructing the LSSC time scheduling model in consideration of the CODP decision problem.

The CODP positioning problem and the time scheduling problem have caught the attention of many researchers. Existing research on the supply chain scheduling model has four deficiencies, which will be discussed in this paper.

First, the conclusions drawn from traditional manufacturing supply chain scheduling models may not be fully applicable to LSSC scheduling problems and it is necessary to consider the characteristics of the service supply chain in the new model. On the one hand, the CODP model of the service supply chain does not include inventory cost since service production is different from manufacturing production. But in the situation of multiorders, a service company also faces the service order pile-up problem in the transition session from the mass service process to the customization service process. Then, how to introduce this cost factor in the objective function? On the other hand, under circumstances of multiorders, the lead times of different service orders are different [7], so are the service processes. Then how to reflect these lead-time constraints and the differences of service processes in the model constraints?

Second, as an important index reflecting supply chain agility, the time requirement of costumers may change in a number of cases (see, e.g., [8, 9]). The operation time requirements to FLSPs may have some variation [7]. Thus, the influence of service completion time on costumers (when service time is ahead of schedule or delayed) or FLSPs on the scheduling results should be considered. When a service is ahead of schedule or delayed, how then is completion time expressed in the model? If the variation of completion time is ahead of time or delayed, what is its effect on supply chain performance?

Third, most of the existing time scheduling models of the supply chain concern only the cost control goal and the time requirement constraint. However, for the LSSC under the MCLS environment, cost control is not always the most

important scheduling goal. The punctual delivery of service orders and satisfaction of FLSPs are also of great importance. Moreover, CODP positioning does not only influence the service operation cost but also the time scheduling effects. Therefore, how to present these objective functions in the model and how can the CODP constraint be reasonably expressed?

Fourth, it is necessary for the LSI managers to know which parameters have an influence on scheduling results. Then in this paper, what scheduling parameters will affect scheduling results? What are the specific influence rules? For the LSI, how to better deal with the time scheduling problem using these influence rules?

The LSSC time scheduling model proposed in this paper solves these four problems mentioned above. To minimize the total order operation costs of the LSSC, minimize the difference between the expected and actual time of completing the service orders, and maximize the satisfaction of functional logistics service providers, this study establishes an LSSC time scheduling model considering the CODP. The results show that the conclusion obtained from our previous study could be extended when considering the CODP decision problem. First, the optimal scheduling table is flexible to some degree; that is, the order completion time of the LSSC is allowed for ahead-of-time and delay within a certain range. As the requests for completing orders in advance become more intensive, the number of optional CODP positions decreases, which leads to either the decrease of the LSSC flexibility or the customization degree of customer orders. Second, the improvement of the supply chain comprehensive performance caused by the increase of the relationship cost coefficient of the logistics service integrator (LSI) is limited. Furthermore, the difference of the LSI's preference for cost and service delivery punctuality leads not only to the difference of the CODP position but also the scheduling performance of the LSSC.

This paper is organized as follows. Section 2 presents the literature review, in which the existing supply chain optimization scheduling models and methods as well as the CODP positioning method are systematically summarized. In Section 3, an LSSC time scheduling model based on the CODP is established. Section 4 presents the model solution, in which a method for solving the multi-objective programming model is provided. Section 5 gives the numerical analysis, in which the influence of relevant parameters on the time scheduling performance is explored. The main conclusions and management insights are discussed in Section 6. The reference value for researchers and managers are given respectively. The last section presents the limitation of this study and provides further research directions in this field.

2. Literature Review

The literature review presents related studies on the time scheduling of the LSSC and the CODP positioning problem under the MCLC environment. Our research direction is proposed after we summarize research development and its deficiencies.

2.1. Mass Customization (MC), CODP, and Postponement Strategy. Since Pine and Davis [10] proposed that the MC mode would become the new frontier in business competition in 1993, after nearly 20 years of development and application, the MC mode has increasingly become a mainstream mode of operation. According to da Silveira et al. [11], MC is the ability to provide customized products or services through flexible processes in high volumes and at reasonably low costs. MC has been extensively studied and applied in the field of manufacturing supply chains because of its significant improvement in operation performance. Many scholars have conducted studies on this topic. Fogliatto et al. [12] conducted a detailed review of the literature on MC production since the 1980s. From the view of the current progress in domestic and international research, research on MC mainly developed in the MC production mode in the manufacturing industry, such as the MC mode and its product development (see, e.g., [13, 14]), production planning and control technology in MC (see, e.g., [15, 16]), costs of MC (see, e.g., [17]), and factors and conditions that influence MC (see, e.g., [18]), among others.

CODP is an important topic in the study of MC production. Yang et al. [19] point out that CODP, as the core element in realizing the MC mode, is one of the effective methods for managing uncertainty. The reasonable positioning of the CODP largely determines the cost and degree of the customization of a supply chain. Therefore, a number of researchers have conducted in-depth studies on the CODP (see, e.g., [20]), from single CODP positioning to multi-CODP positioning (see, e.g., [21, 22]) and from static research to dynamic research (see, e.g., [23]).

Postponement is closely related to CODP. The concept "postponement" was first introduced by Alderson [24] in his paper *Marketing efficiency and the principle of postponement.* He defined postponement as a marketing strategy of putting off the changes in forms and features as much as possible. Postponement manufacturing strategy has been widely used in supply chain researches. Shapiroe [25] studied the postponement strategy positioning problem and established the correlation between the supply chain and the postponement strategy. According to the difference in the degree of customization requested by customers, Bowerson and Closs [26] divided the postponement strategies into three categories, which are Time Postponement, Place Postponement, and Form Postponement. Larry et al. [27] developed a new methodology for analyzing the impact of forecast accuracy on the decision to postpone production. In the implement of postponement manufacturing, the CODP position is critical. By changing the CODP position, postponement degree is increased or decreased, and then accordingly the degree of customization is adjusted. On the one hand, enabling manufacturing postponement can provide firms with a prompt response [28]. On the other hand, postponement has recently been mentioned as a useful tool for managing supply risk and disruptions [29].

In recent years, studies on CODP in the service industry have become a research focus. Since the CODPs in service industry and manufacturing industry share similarities and show differences (as mentioned in Section 1), some researchers have already applied CODP theory from the

manufacturing industry to the service industry. For example, Tang and Chen [30] applied postponement theory originated from manufacturing to operational management in the service industry and discussed the factors that needed to be considered in CODP positioning. However, the study was limited to a single service enterprise. Specialized research on the CODP for a general service supply chain, especially for LSSC, is lacking. Therefore, considering the features of the service supply chain, discussing the CODP positioning problem in the service supply chain, and analyzing the effect of CODP on supply chain scheduling performance are necessary.

2.2. Time Scheduling of Supply Chain. Most previous research on supply chain scheduling focused on the manufacturing industry and achieved substantial results. In 2003, Hall and Potts [31] published "*Supply chain scheduling: batching and delivery*," an early systematically research on the supply chain scheduling model. Many earlier studies on supply chain scheduling paid attention to the job shop scheduling within a single enterprise, such as Lee et al. [32] and Philipoom [33]. The main concern of these studies is the arrangement of processing procedures and the order operation sequence. Some scholars studied assembly system coordination in manufacturing enterprises (see, e.g., [34]). Studies on supply chain scheduling in the MC production mode have emerged in recent years (see, e.g., [35, 36]).

In terms of supply chain time scheduling, some scholars conducted dominant contradiction analysis and studied supply chain scheduling optimization solutions in MC (see, e.g., [37]). Apparently, some differences exist between the ideal scheduling timetables of different supply chain members and those of the customer demand. Dawande et al. [38] explored ways to solve this discrepancy, consequently inspiring us to conduct our research. Based on the integration of supply chain production planning and the scheduling process, Mishra et al. [39] designed a mixed integer programming model. Similar to the CODP positioning problem, cost is the primary factor considered in supply chain scheduling (see, e.g., [40, 41]). Most studies assume that the order completion time required by customers or the delivery time required by suppliers is fixed. However, as important index reflecting supply chain agility, the time requirements of customers may change in a number of cases (see, e.g., [8, 9]). Moreover, the operation time requirements to FLSPs do not have strict limitations, enabling a certain amount of variation [7]. The influence of service completion time, whether ahead of schedule or delayed by customers or FLSPs, on the scheduling results should be considered. Aside from the cost objective, the punctual delivery of service orders and the satisfaction of FLSPs directly influence customer satisfaction as well. Therefore, considering the influence of the differences among the degrees of importance of the objective functions on supply chain performance is necessary. However, the current literature has not addressed this issue.

Although research on supply chain scheduling in the MC environment has increasingly improved, studies on the service supply chain remain significantly insufficient. Similar to that on the manufacturing supply chain, research on the service supply chain mainly focuses on service process scheduling (see, e.g., [30]) and order assignment scheduling (see, e.g., [4, 42]), among others. However, only a few studies have been made on time scheduling, which is an important part of LSSC scheduling. Carrying out research on time scheduling combined with a service supply chain (especially LSSC) in certain industries is necessary.

From the review of relevant literature, it can be concluded that specialized research on CODP for service supply chain, especially for LSSC, is still insufficient as well as research on the time scheduling of LSSC. Under the MCLS mode, the time scheduling of LSSC should consider the influence of CODP on scheduling results. Furthermore, we find that three important issues remain unresolved. First, under the MCLS environment, how to introduce the mass characteristics and the personalized characteristics of the customization service? Second, in the case of multiservice orders, the lead-times of different service orders are different [7], so are the service processes. How to express the lead time constraints of customer orders? How to demonstrate the difference of the service process required by customer orders in the constraints? Third, the scheduling model always has multiple objective functions in which time objective (the punctual service delivery objective) and the cost objective coexist. The relative concern degree of LSI on cost and service delivery punctuality may change in different environments. Thus, what is the effect of these changes on scheduling performance? These three problems are discussed in the Model Building Section of this paper.

3. Model Building

In this section, the LSSC time scheduling model considering CODP in the case of multiservice orders is established. This scheduling model is a multiobjective programming model since in the scheduling process, the LSI needs to consider three scheduling objectives, which are minimizing the total order operation costs of the LSSC, minimizing the difference between the expected and actual time of completing the service orders, and maximizing the satisfaction of functional logistics service providers. In addition, some constraints need to be met in this scheduling model, including the completion time constraints required by customers, the time correlation constraint between the upstream process and the downstream one, and the satisfaction constraint.

In Section 3.1, we describe the problem and the sequence of events involved in the model. In Section 3.2, the important model assumptions are proposed. In Section 3.3, the LSSC time scheduling model, which is a multiobjective programming model, is presented.

3.1. Problem Description. A two-echelon LSSC composed of one LSI and many FLSPs are assumed. The LSI handles multiple service orders from customers at the same time. Each logistics service order consists of multiple service processes, which are divided into two types, namely, the customized service process and the mass service process. The mass service process of customer i is conducted either it is

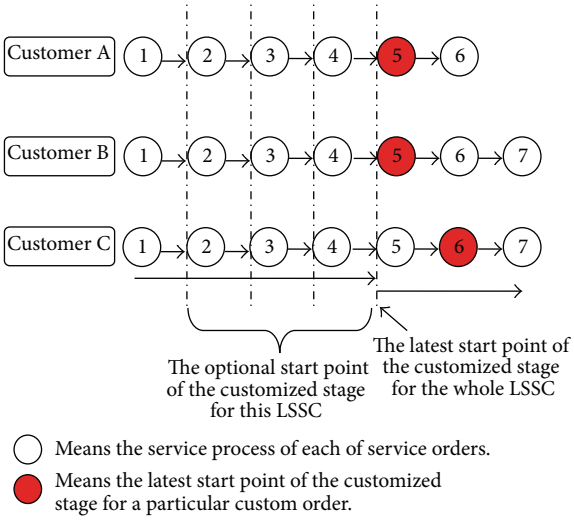


Figure 1: Schematic diagram of each order's operation process requirement in numerical example.

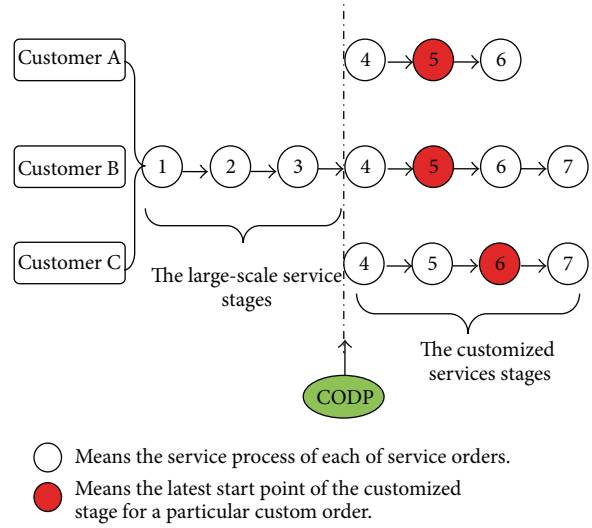


Figure 2: Take CODP is $k = 4$ for example, the orders' operation processes schematic diagram.

integrated into the mass service process of customer $j(j \neq i)$ for a consolidated mass service operation or it is operated independently in its customized mode. The LSI analyses all these orders' service processes and their respective time requests and then inquires the FLSPs of each service process about its standard time for completing the service process. After that, the LSI needs to determine the optional CODP collection. Then the LSI sets up the scheduling optimization goals based on scheduling time, scheduling costs, and satisfaction degree of FPLSs, and chooses the optimal CODP position, obtaining the schedule plan.

We use an example to demonstrate the scheduling problem. For example, as shown in Figure 1, there are three service orders from three customers that need to be scheduled by the LSI. The total amount of processes and the completion time of each order are different. However, some processes can be carried out together in the MC mode because of the similarity of the order contents. Assume that the total number of the service processes for these three orders is 6, 7, and 7, respectively, and these orders must be operated in the customized mode from the 5th, 5th, and 6th process according to customer requirements. Thus, without considering other constraints, the optional CODP in this numerical example is one of the elements in $\{2, 3, 4, 5\}$. Figure 2 shows the LSSC order operation schematic when the CODP is located in Process 4.

The sequence of events in the LSSC time scheduling is shown in Figure 3, which consists of eight steps to complete the entire time scheduling.

(1) J customers' orders arrive and each order has its own customized requirements.

(2) The LSI analyzes the service processes and requirements of each service order (and distinguishes the processes that can be operated in mass service mode from the ones that must be operated in the customized mode).

(3) The LSI inquires the FSLPs about their standard completion time for each process.

(4) The LSI decides the collection of possible CODPs.

(5) Systematically considering three scheduling objectives, namely, the optimal total scheduling time, total costs of scheduling, and the satisfaction of FLSPs, the LSI establishes the time scheduling model and decides the optimal CODP and then obtains the complete specific scheduling plan.

(6) According to the scheduling plan, FLSPs deploy their capability to guarantee that the completion time request to be met.

(7) The FLSPs offer corresponding service according to the LSI's scheduling plan (maybe mass service or customization service).

(8) Customer orders are finished.

In this study, by solving the model in step 5, the optimal CODP is determined and the scheduling plan is output. Based on this, we conduct a further analysis to explore the effect of changes in various scheduling parameters on the CODP positioning and the supply chain performance. Table 1 shows the model notations.

3.2. Model Assumptions

Assumption 1. The LSI and FLSPs have established cooperative relationship and built the LSSC through contracts in strategic level. This paper focuses on how to allocate operation time for different service processes under circumstances where the cooperative relationship has been decided in order to maximize the supply chain scheduling performance. This paper does not consider the contract coordination problem in strategic level.

Table 1: Notations for the model.

Notations	Description
c	The relationship cost coefficient of the LSI.
C_i	The normal service cost per unit time per unit quantity of the ith process in offering mass operation.
C_i^{ext}	The extra service cost per unit time per unit quantity of the ith process in offering mass operation.
C_{ij}	The normal service cost per unit time per unit quantity of the ith process in offering customized operation for the jth customer.
D_i	In mass processes, the penalty cost per unit time per unit quantity of the ith process, if the order completion time is delayed.
D_{ij}	In customized processes, the penalty cost per unit time per unit quantity of the ith process in offering customized operation for the jth customer, if the order completion time is delayed.
F_{ij}	If the ith process is chosen to be CODP, the cost unit time per unit quantity needs to pay for the jth customer order.
H_{ij}	The operation and switching time in the ith service process for the jth customer order due to the switching from mass to customized operation. It varies with i and j and is decided by the technology capability of the FLSP in the ith process.
I_j	The total amount of service processes of the jth customer order.
k	The optimal CODP for all the orders.
K_j	The latest possible CODP for the jth customer order (the latest point where customized operation must begin).
k_t	Weight of the objective function Z_t in Z', $t = 1, 2, 3$.
K_t	Weight of the objective function Z_t in Z, $t = 2, 3$.
ΔK_j	The difference between the latest possible CODP for the jth customer order and the optimal CODP for all the orders, $\Delta K_j = K_j - k$.
L_j	The total amount of processes in the jth customer order.
N	The total amount of customers' orders.
P_i	In mass processes, the penalty cost per unit time per unit quantity of the ith process, if order is finished ahead of the expected time.
P_{ij}	In customized processes, the penalty cost per unit time per unit quantity of the ith process in offering customized operation for the jth customer, if order is finished ahead of the expected time.
R	The coefficient of relative concern degree of LSI for cost and service delivery punctuality.
T_i	For a certain order set, the normal service time of the ith process in offering mass operation, $i = 1, 2, 3, \ldots, I_0$, the same below.
T_{ij}	The normal service time of the ith service process in offering customized operation for the jth customer, $j = 1, 2, 3, \ldots, J_0$, the same below.
T_i^{exp}	The expected operation time of FLSPs for the ith service process set by LSI in offering mass operation.
T_j^{exp}	The expected completion time of an order set by LSI's jth customer.
T_{ij}^{exp}	The expected operation time of FLSPs for the ith service process in offering customized operation for the jth customer set by LSI.
T_{i+1}^{+}	In mass processes, the upper limit of the time delay incurred in the $(i-1)$th service process which could be endured by the ith service process. It is determined by the rigid requirement caused by upstream and downstream operations of LSSC.
T_{i+1}^{-}	In mass processes, the upper limit of the time ahead of schedule incurred in the $(i-1)$th service process which could be endured by the ith service process, which is determined by the rigid requirement caused by upstream and downstream operations of LSSC.
$T_{(i+1),j}^{+}$	In customized processes, for the jth customer order, the upper limit of the time delay incurred in the $(i-1)$th service process which could be endured by the ith service process, which is determined by the rigid requirement caused by upstream and downstream operations of LSSC.
$T_{(i+1),j}^{-}$	In customized processes, for the jth customer order, the upper limit of the time ahead of schedule incurred in the $(i-1)$th service process which could be endured by the ith service process, which is determined by the rigid requirement caused by upstream and downstream operations of LSSC.
U_i^0	The lower limit of the satisfaction degree of the ith mass process.
U_{ij}^0	The lower limit of the satisfaction degree of the ith customized process of the jth customer order.
Y_j	The number of customized orders of the jth customer.
Z_1	The total cost of LSSC.
Z_2	The closeness degree of the actual order completion time and the expected one set by its customer.
Z_3	The average satisfaction of all processes in LSSC.
Z	The objective function synthesized by Z_2 and Z_3, which is also called the comprehensive performance of LSSC.
Z^*	The optimal value of Z.
$Z_1^{\min}$	The minimum of Z_1 when not considering the objective functions Z_2 and Z_3.
Z'	The objective function synthesized by Z_1, Z_2, and Z_3.
Z'^*	The optimal value of Z'.
β	The delay coefficient of the order completion time permitted by its customer.
ρ	The coefficient of mass service effects obtained by LSI, which presents the cost reduction due to the increase of mass operations.
w	The customer order difference tolerance coefficient of LSI.

Table 1: Continued.

Notations	Description
θ	The coefficient of the order differences, which reflects the degree of differences among each order in terms of customized degree in a certain order set. $\theta = (1/J_0)\sum_{j=1}^{J_0}(\Delta K_j/K_j)$ $i = 1, 2, 3, \ldots, I_0,\ j = 1, 2, 3, \ldots, J_0$

Note: k, T_i^{ext}, and T_{ij}^{ext} are decision variables.

Assumption 2. Generally, the LSI possesses strong integration capability. It normally has a number of FLSPs and utilizes their capacities to meet different kinds of logistics service requests from customers. Therefore, we assume that logistics service capacities in each process are adequate and thus there is not any capacities constraint. Furthermore, we assume that the logistics service in each process is completed by one FLSP. The FLSPs can provide either mass service or customization service. But the cost of offering mass service is lower than that of offering customized service, and the normal service time to finish an order in the mass service mode is longer than that in the customized mode.

Assumption 3. In this paper, we assume that the normal operation time of each provides is settled and stationary, which is decided by the service order.

Assumption 4. Each scheduling task aims at only one set of customer orders and no new orders are added.

Assumption 5. In terms of time scheduling, additional service costs are incurred, regardless of whether the order is completed ahead of time or delayed. For the jth customer's order in the ith process, regardless of whether the order is completed ahead of time or delayed, the unit additional service cost C_{ij}^{ext} is assumed to be the same.

Assumption 6. The normal service time refers to the usual time needed in completing a task using its FLSP capability. When the work is done in normal service time, the satisfaction of the FLSP is the highest. Conversely, if the FLSP works in abnormal time (e.g., time ahead of schedule or delayed), its satisfaction will decline.

Assumption 7. Based on the premise of meeting the customized requirements, each node (except at the beginning and the end of the logistics services) can be used as the CODP.

Assumption 8. No matter which step is chosen to be the CODP, switching operation (i.e., loading and unloading) time and cost exists. The operation time and unit cost are different because of the difference in CODP positioning.

3.3. Model Building

3.3.1. Optimization Objectives of the Scheduling Model. When the LSI undertakes the time scheduling process of LSSC, multiple factors should be considered systematically. For example, the LSI needs to consider multiple requirements in terms of the service order's completion time and attempt to finish the service on time. The LSI cannot ignore the factor of the satisfaction of all FLSPs and must arrange transfer between each service process reasonably. To achieve the minimal scheduling cost, it is better to operate the service processes in mass service mode as more as possible. Therefore, to minimize the total cost of LSSC orders, minimize the difference between the expected time and actual time of completing the service orders, and maximize the satisfaction of all FLSPs, this paper establishes the LSSC time scheduling model based on the CODP. On the premise that the FLSPs' supply capacities are with uncertainty, the decision variable of the LSI to implement the LSSC time scheduling is the expected actual completion time of each FLSP. The FLSPs will adjust their completion time (compress or delay the order completion time) according to the difference between the expected time and their own normal completion time to reach the optimal scheduling performance.

$$\text{Min} \quad Z_1 = f_1 + f_2 + f_3 + f_4 + f_5, \tag{1}$$

$$\begin{aligned} \text{Min} \quad & Z_2 = \sum_{j=1}^{J_0} \frac{\left|T_j^{\text{exp}} - T_j\right|}{T_j^{\text{exp}}} \times \frac{Y_j}{N}, \\ & T_j = \sum_{i=1}^{I_0} T_{ij} + T_{ij}^{\text{ext}} + T_i + T_i^{\text{ext}}, \end{aligned} \tag{2}$$

$$\begin{aligned} \text{Max} \quad & Z_3 \\ & = \left(\sum_{i=1}^{k-1}\left(1 - \frac{\left|T_i - T_i^{\text{exp}}\right|}{T_i}\right)\left(\frac{T_i C_i}{T_i C_i + \left|T_i^{\text{ext}}\right| C_i^{\text{ext}}}\right)\right. \\ & \quad + \sum_{j=1}^{J_0}\sum_{i=k}^{I_0}\left(1 - \frac{\left|T_{ij} - T_{ij}^{\text{exp}}\right|}{T_{ij}}\right) \\ & \quad \left.\times\left(\frac{T_{ij} C_{ij}}{T_{ij} C_{ij} + \left|T_{ij}^{\text{ext}}\right| C_{ij}^{\text{ext}}}\right)\right) \\ & \quad \times\left(k - 1 + \left(\sum_{j=1}^{J_0}\left(I_j - (k-1)\right)\right)\right)^{-1}, \end{aligned} \tag{3}$$

where

$$[f(x)]^+ = \max\{0, f(x)\},$$

$$f_1 = (1 - \rho k)\sum_{i=1}^{k-1}\left(T_i C_i + \left|T_i^{\text{ext}}\right| C_i^{\text{ext}}\right) \times Y,$$

$$f_2 = \sum_{j=1}^{J_0}\sum_{i=k}^{I_j}\left(T_{ij} C_{ij} + \left|T_{ij}^{\text{ext}}\right| C_{ij}^{\text{ext}}\right) \times Y_j,$$

$$f_3 = \sum_{j=1}^{J_0} F_{ij} H_{ij} Y_j,\ \ i = k,$$

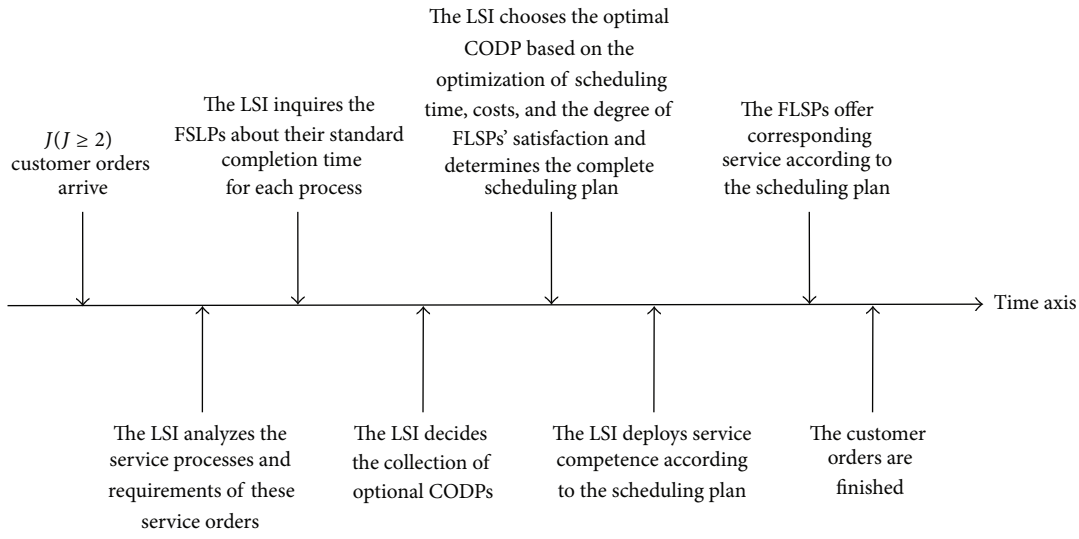


Figure 3: Sequence of events in the time scheduling of the LSSC.

$$f_4 = \sum_{i=1}^{k-1} \left[T_i^{\exp} - T_i - T_i^{\text{ext}}\right]^+ P_i \times Y + \sum_{j=1}^{J_0} \sum_{i=k}^{I_j} \left[T_{ij}^{\exp} - T_{ij} - T_{ij}^{\text{ext}}\right]^+ P_{ij} \times Y_j,$$

$$f_5 = \sum_{i=1}^{k-1} \left[T_i + T_i^{\text{ext}} - T_i^{\exp}\right]^+ D_i \times Y + \sum_{j=1}^{J_0} \sum_{i=k}^{I_j} \left[T_{ij}^{\exp} - T_{ij} - T_{ij}^{\text{ext}}\right]^+ D_{ij} \times Y_j. \quad (4)$$

In (1), the objective function Z_1 is used to minimize the total scheduling cost of LSSC. f_1 is the total cost of all costs incurred in the mass service processes. $(1 - \rho k)$ denotes the influence caused by the mass service effects. Along with the increase of mass service processes k, the unit cost of mass service decreases. f_2 is the total cost of all costs incurred in the customized processes. f_3 is the switching cost in CODP, which refers to the switching operation cost incurred when the mass service operation is completed and the customized operation is to be started. f_4 is the penalty cost for the process, which is completed earlier than the expected time. f_5 is the penalty cost for the process that is delayed compared with the expected completion time.

In (2), the objective function Z_2 is used to complete all of these orders as punctually as possible to minimize the difference between the actual completion time and the expected completion time. Each weight is the proportion of each customer's order to the entire order.

In (3), the objective function Z_3 is used to maximize the satisfaction of all FLSPs. The first part in the equation is the sum of satisfaction in the mass service operation stage, and the second part is the sum of satisfaction in the customized operation stage. Both parts are represented by the product of satisfaction in terms of time and satisfaction in terms of cost. $(1 - (|T_{ij} - T_{ij}^{\exp}|/T_{ij}))$ indicates the proximity degree between the normal operation time and the order expected time set by the LSI for the jth FLSPs in the ith service process, which represents satisfaction in the time aspect. $T_{ij}C_{ij}/(T_{ij}C_{ij} + |T_{ij}^{\text{ext}}|C_{ij}^{\text{ext}})$ indicates the proportion of the normal operation cost in the total cost of the jth customer's order in the ith service process, which represents the FLSP's satisfaction in the cost aspect. The denominator of (3) is the total amount of service processes. In the mass service stage, each service process carried out for all orders counts as one process; in the customized stage, each process for a single order counts as one process.

3.3.2. Constraints of the Scheduling Model. The scheduling model needs some constraints, including the time constraint of the order completion time required by customers, the time constraint of the upstream process and downstream process, and the FLSPs' satisfaction degree constraint. Equations (5) to (10) are the constraints of the model:

$$\text{subject to} \quad \sum_{i=1}^{k-1} \left(T_i + T_i^{\text{ext}}\right) + \sum_{i=k}^{I_j} \left(T_{ij} + T_{ij}^{\text{ext}}\right) + H_{kj} \leq T_j^{\exp} \left(1 + \beta\right), \quad (5)$$

$$T_{i+1}^- \leq T_i + T_i^{\text{ext}} - T_i^{\exp} \leq T_{i+1}^+, \quad i \leq k-1, \quad (6)$$

$$T_{(i+1)j}^- \leq T_{ij} + T_{ij}^{\text{ext}} - T_{ij}^{\exp} \leq T_{(i+1)j}^+, \quad i \geq k, \quad (7)$$

$$\left(1-\frac{\left|T_i-T_i^{\exp}\right|}{T_i}\right)\left(\frac{T_iC_i}{T_iC_i+\left|T_i^{\text{ext}}\right|C_i^{\text{ext}}}\right) \geq U_i^0, \quad i \leq k-1, \tag{8}$$

$$\left(1-\frac{\left|T_{ij}-T_{ij}^{\exp}\right|}{T_{ij}}\right)\left(\frac{T_{ij}C_{ij}}{T_{ij}C_{ij}+\left|T_{ij}^{\text{ext}}\right|C_{ij}^{\text{ext}}}\right) \geq U_{ij}^0, \quad i \geq k, \tag{9}$$

$$\theta=\frac{1}{J_0}\sum_{j=1}^{J_0}\frac{\Delta K_j}{K_j}\leq\omega. \tag{10}$$

Equation (5) is the time constraint of the customer requirements, which means that the completion time of each order cannot be longer than the maximum time delay range set by the corresponding customer. Equation (6) is the time constraint due to the connection relation between upstream and downstream in the mass service stage. This strong constraint must be obeyed because in the service provision process, the link in operation time exists between the upstream and downstream service processes. Equation (7) is the time constraint due to the connection between upstream and downstream in the customized stage. It is also a strong constraint. Equations (8) and (9) are the constraints of the FLSPs' satisfaction in the mass service stage and the customized stage, respectively, indicating that the satisfaction of each FLSP should be more than the lower limit that they can accept. Equation (10) is the constraint of order differences, which indicates that the actual order difference coefficient cannot be more than the one set by the LSI.

4. Model Solution

The solution of this LSSC time scheduling model is presented in this section. Generally, researches on scheduling are mainly categorized into two branches. One branch is aimed at studying the effectiveness and optimality in order to carry out quick scheduling in practice; another branch of the research focus is studying the influence of relative parameters on scheduling performance to find out the most influential ones and determine them reasonably. In Introduction, we points out that this paper mainly works on four aspects of deficiency in existing researches. Among them, the fourth kind of deficiency is discussing what scheduling parameters will affect scheduling results. Therefore, the model solution does not focus on the comparison or selection of different kinds of solution method instead it just chooses an appropriate solution method. In Section 4.1, this multiobjective model is simplified into a single-objective programming model. In Section 4.2, the genetic algorithm is used to solve this simplified model.

4.1. Simplifying the Multiobjective Programming Model. The LSSC time scheduling model has three objectives and six constraints. It is a typical multiobjective programming problem. The model solution cannot be simply carried out from the perspective of mathematical equations; its practical meaning should be considered as well. In the MCLS mode, cost is not the only consideration. Completing the order with the absolutely minimal cost is not necessary. It only needs to maintain the total cost of the LSSC within a certain range. To build and maintain a good relationship with FLSPs, LSIs are usually willing to give a certain amount of cost concession. Thus, the cost objective may be transferred to a new constraint (see, e.g., [43]). Taking these actual situations into account, we introduce a parameter called the relationship cost coefficient c into the model and use it to represent the cost augment limits. The new constraint of cost is shown in

$$Z_1 < Z_1^{\min}\times(1+c). \tag{11}$$

The new constraints are the original constraints combined with (11).

The original model then becomes a twin goal programming problem whose objectives are customer service time and satisfaction of FLSPs, that is, minimal service delivery time and maximum FLSP satisfaction. As some conflicts and incommensurability exist in each target in multiobjective decision-making problems, finding an absolute optimal solution is difficult. In terms of the solving method for multiobjective programming problems, many specialized solution methods can be used, such as the evaluation function method (e.g., linear weighting method, reference target law, and minimax method), target planning method, hierarchical sequence law, interactive planning, and subordinate function method. In this paper, referring to Liu et al. [6] and Liu et al. [43], we choose the most typical linear weighting method to solve our model. The objective function Z_2 is used to find the minimal value of Z_2, and the objective function Z_3 is used to find the minimal value of Z_3, both of which are dimensionless and $Z_2 \in (0,1)$, $Z_3 \in (0,1)$. After the mathematical transformation, the synthesized objective function is shown in

$$\max Z = K_2\times(1-Z_2)+K_3\times Z_3. \tag{12}$$

In (12), K_2 and K_3, represent the weights of Z_2 and Z_3, respectively, which are determined through the linear weighting method (see, e.g., [43]). The new objective denotes the synthesized effect of the LSSC time scheduling, which is called the comprehensive performance of the LSSC.

The solving method in (12) also accords with the actual LSSC scheduling process. When implementing the multi-order scheduling process, the LSI needs to meet multicustomer completion requests as well as the satisfaction degree of FLSPs. If the LSI excessively emphasizes meeting the customers' time requests, its FLSPs may be unsatisfied which would lead to failure in the order completion. On the contrary, if the LSI immoderately emphasizes improving FLSPs' satisfaction, the service time of customer orders may not be guaranteed. Therefore, in the actual scheduling process, the LSI needs to keep a good balance between the objective weights of Z_2 and Z_3, trying to maximize the comprehensive performance objective of the supply chain. In the numerical analysis, Sections 5.6.1 and 5.6.2 will demonstrate the influence of different weights of objectives on the scheduling results.

Thus, the single-objective model is as follows:

$$\max \quad Z = K_2 \times (1 - Z_2) + K_3 \times Z_3, \tag{13}$$

$$\text{subject to} \quad \sum_{i=1}^{k-1} \left(T_i + T_i^{\text{ext}}\right) + \sum_{i=k}^{I_j} \left(T_{ij} + T_{ij}^{\text{ext}}\right) + H_{kj} \le T_j^{\text{exp}} (1 + \beta), \tag{14}$$

$$T_{i+1}^{-} \le T_i + T_i^{\text{ext}} - T_i^{\text{exp}} \le T_{i+1}^{+}, \quad i \le k-1, \tag{15}$$

$$T_{(i+1)j}^{-} \le T_{ij} + T_{ij}^{\text{ext}} - T_{ij}^{\text{exp}} \le T_{(i+1)j}^{+}, \quad i \ge k, \tag{16}$$

$$\left(1 - \frac{\left|T_i - T_i^{\text{exp}}\right|}{T_i}\right)\left(\frac{T_i C_i}{T_i C_i + \left|T_i^{\text{ext}}\right| C_i^{\text{ext}}}\right) \ge U_i^0, \quad i \le k-1, \tag{17}$$

$$\left(1 - \frac{\left|T_{ij} - T_{ij}^{\text{exp}}\right|}{T_{ij}}\right)\left(\frac{T_{ij} C_{ij}}{T_{ij} C_{ij} + \left|T_{ij}^{\text{ext}}\right| C_{ij}^{\text{ext}}}\right) \ge U_{ij}^0, \quad i \ge k, \tag{18}$$

$$\theta = \frac{1}{J_0} \sum_{j=1}^{J_0} \frac{\Delta K_j}{K_j} \le \omega, \tag{19}$$

$$Z_1 \le Z_1^{\min} \times (1 + c). \tag{20}$$

4.2. Using the Genetic Algorithm to Solve the Multiobjective Programming Problem. The genetic algorithm is an effective method used to search for the optimal solution by simulating the natural selection process. As it uses multiple starting points to begin the search, it has a satisfactory global search capability. Since its overall search strategy and optimization search method are not dependent on gradient information or other aids but only the objective function, which affects search direction and corresponding fitness functions, the genetic algorithm offers an common framework to solve complicated system problems. The genetic algorithm does not depend on a specific field and is robust to the kinds of problems. Thus, genetic algorithm is widely used in many scientific fields today, such as combinatorial optimization (see, e.g., [44]), machine learning (see, e.g., [45]), signal processing (see, e.g., [46]), adaptive control, and artificial life (see, e.g., [47]). For the combinatorial optimization problem, the genetic algorithm is quite effective to solve NP problem, such as the production scheduling problem (see, e.g., [48–50]), travelling salesman problem (see, e.g., [51]), knapsack problem (see, e.g., [52]), bin packing problem (see, e.g., [53]), and graph partitioning problem (see, e.g., [54]).

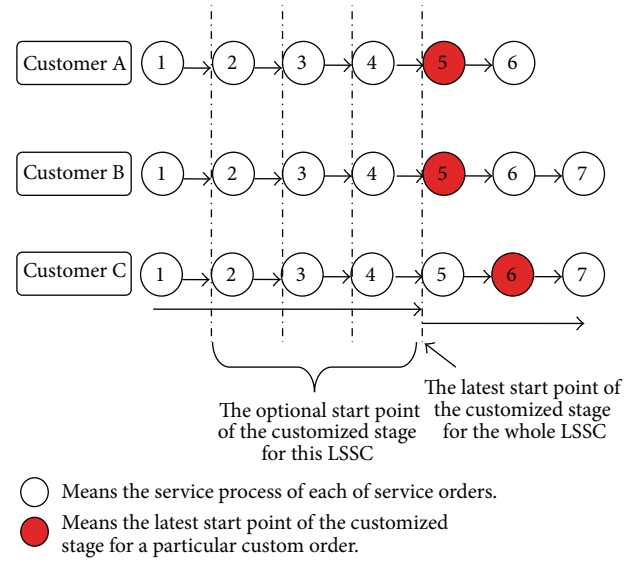


Figure 4: Schematic diagram of each order's operation process requirement in numerical example.

Similar to the natural evolutionary processes, the computational process of the genetic algorithm is an iterative process that involves selection, crossover, and mutation processes. This kind of successive inheritance from individuals with high fitness to the next generation obtains the optimal solution at last. Our solution to the scheduling model does not focus on comparing or selecting the best method among different kinds of solution methods, and we just choose an appropriate method. Given the superiority of the genetic algorithm in solving programming problems and the successful application to scheduling problems (see, e.g., [48–50]), this paper uses the genetic algorithm to solve the proposed model.

5. Numerical Analysis

This section illustrates the validity of model by conducting a numerical analysis and then by exploring the influence of relevant parameters on the time scheduling results. We also give some effective recommendations for supply chain scheduling and optimization based on numerical analysis. Section 5.1 presents the basic data of the numerical example. Section 5.2 shows the scheduling results. Section 5.3 discusses the influence of the time delay coefficient β of order completion on the scheduling results of the LSSC. Section 5.4 presents the influence of the relationship cost coefficient c on the scheduling results. The influences of the order difference tolerance coefficient w and the mass service effects coefficient ρ obtained by LSI on the scheduling results are given in Sections 5.5 and 5.6, respectively.

5.1. Numerical Example Description and Basic Data. See from Figure 4 a two-echelon LSSC is assumed to consist of

Table 2: Basic data.

Parameter	i						
	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
T_i	10	9	10	15	8	12	8
C_i	4	3	5	7	5	2	3
C_i^{ext}	6	5	7	9	7	5	5
T_i^{exp}	13	8	7	13	10	8	12
P_i	7	5	10	12	7	9	6
D_i	4	3	8	10	5	7	5
U_i	0.5	0.5	0.6	0.55	0.55	0.4	0.5
T_{i+1}^{+}	—	3	4	3	3.5	5	3.5
T_{i+1}^{-}	—	−3	−4	−5	−5	−4	−4

one LSI and many FLSPs. The LSI handles three customer service orders at the same time. The total process amount and completion time of each order are different. However, some processes can be carried out together in the MC mode because of the similarity of the order contents. The total number of service processes for these three orders is 6, 7, and 7, respectively. These orders must be operated in the customized mode from the 5th, 5th, and 6th processes according to customer requirements. Thus, without considering other constraints, the optional CODP in this numerical example is one of the elements in $\{2, 3, 4, 5\}$.

The parameter values used in our model are shown in the matrix below and in Table 2:

$$T = \begin{bmatrix} 7 & 6 & 6.5 \\ 6 & 11 & 7 \\ 10 & 9 & 9.5 \\ 12 & 10 & 11 \\ 6 & 7 & 7.5 \\ 6 & 7 & 8 \\ — & 7 & 10 \end{bmatrix}, \quad C = \begin{bmatrix} 5 & 6 & 5.5 \\ 7 & 8 & 8.5 \\ 11 & 12 & 10.5 \\ 15 & 18 & 14.5 \\ 6 & 5 & 7.5 \\ 7 & 6 & 7 \\ — & 4 & 8 \end{bmatrix},$$

$$F = \begin{bmatrix} 8 & 7 & 7 \\ 8 & 6 & 6 \\ 7 & 6 & 5 \\ 5 & 4 & 4 \\ 4 & 3 & 3 \\ 4 & 3 & 2 \end{bmatrix}, \quad H = \begin{bmatrix} 3 & 2.5 & 2.5 \\ 3 & 2 & 2 \\ 2.5 & 2 & 2 \\ 4 & 4 & 3 \\ 5 & 3 & 6 \\ 5 & 6 & 7 \end{bmatrix},$$

$$T^{\text{exp}} = \begin{bmatrix} 9 & 7 & 9 \\ 9 & 8 & 5 \\ 8 & 9 & 13 \\ 10 & 13 & 10 \\ 8 & 6 & 9 \\ 5 & 8 & 12 \\ — & 9 & 12 \end{bmatrix}, \quad P = \begin{bmatrix} 4 & 6 & 6 \\ 6 & 8 & 6 \\ 8 & 8 & 10 \\ 10 & 7 & 5 \\ 6 & 7 & 5 \\ 6 & 8 & 7 \\ — & 4 & 5 \end{bmatrix},$$

$$D = \begin{bmatrix} 2 & 4 & 4 \\ 5 & 4 & 4 \\ 7 & 8 & 9 \\ 8 & 8 & 5 \\ 5 & 4 & 5 \\ 6 & 5 & 4 \\ — & 5 & 3 \end{bmatrix}, \quad C^{\text{ext}} = \begin{bmatrix} 7 & 8 & 7 \\ 10 & 9 & 10 \\ 13 & 14 & 13 \\ 18 & 20 & 16 \\ 8 & 7 & 9 \\ 9 & 8 & 9 \\ — & 6 & 10 \end{bmatrix},$$

$$U = \begin{bmatrix} 0.6 & 0.5 & 0.55 \\ 0.5 & 0.6 & 0.55 \\ 0.4 & 0.5 & 0.55 \\ 0.4 & 0.4 & 0.65 \\ 0.5 & 0.45 & 0.5 \\ 0.6 & 0.6 & 0.5 \\ — & 0.5 & 0.65 \end{bmatrix},$$

$$T_{(i+1),j}^{-} = \begin{bmatrix} — & — & — \\ -3 & -3 & -5 \\ -4 & -4 & -3 \\ -5 & -2.5 & -5 \\ -5 & -4 & -3 \\ -4 & -5 & -2.5 \\ — & -2.5 & -4 \end{bmatrix},$$

$$T_{(i+1),j}^{+} = \begin{bmatrix} — & — & — \\ 3 & 4 & 5 \\ 4 & 5.5 & 3 \\ 3 & 4 & 4 \\ 3.5 & 5 & 2 \\ 5 & 3.5 & 4.5 \\ — & 5 & 3 \end{bmatrix}. \tag{21}$$

Considering its practical significance, our numerical example requires that

$$\left|T_{ij}^{\text{ext}}\right| \le 0.3T_{ij}, \qquad \left|T_i^{\text{ext}}\right| \le 0.3T_i. \tag{22}$$

As shown in (22), in practical scheduling, FLSP's completion time range of a certain service process is generally proportional to the normal working time, regardless of whether the operation time is delayed or ahead of schedule. In this paper, we assume that a certain service process is delayed or completed in advance by not more than 0.3 times the normal working time.

5.2. Numerical Example Results. In the model solution, we use the genetic algorithm and Matlab 7.8 software to solve the problem. Assuming that the genetic population is 800, the hereditary algebra is 800, the delay coefficient of the order completion time is $\beta = 0.05$, the relationship cost coefficient is $c = 0.2$, the order difference tolerance coefficient is $w = 0.5$,

Table 3: Results of numerical example calculation.

Service stage	Service process		Normal service time	Additional service time	Actual service time
Mass stage	Process 1		10	0.0759	10.0759
	Process 2		9	0.2400	9.2400
	Process 3		10	−0.0929	9.9071
Customized stage	Process 4	Customer 1	12	1.4879	13.4879
		Customer 2	10	0.7492	10.7492
		Customer 3	11	−0.9107	10.0893
	Process 5	Customer 1	6	1.4837	7.4837
		Customer 2	7	0.0022	7.0022
		Customer 3	7.5	0.0113	7.5113
	Process 6	Customer 1	6	0.6990	6.6990
		Customer 2	7	0.0155	7.0155
		Customer 3	8	0.0102	8.0102
	Process 7	Customer 2	7	0.0000	7.0000
		Customer 3	10	−0.8458	9.1542

and the mass service effect coefficient obtained by LSI is $\rho = 0.1$, the calculation result is as follows.

The optimal solution is $Z = 0.9324$, and $k = 4$ is the optimal CODP. Thus, the first three processes are part of the mass service operation stage, and the remaining ones are part of the customized operation stage. The corresponding scheduling results are as follows:

Mass service operation stage:

$$\left[T_1^{\text{ext}} \quad T_2^{\text{ext}} \quad T_3^{\text{ext}}\right] = \left[0.0759 \quad 0.2400 \quad -0.0929\right]. \tag{23}$$

Customized operation stage:

$$\begin{bmatrix} T_{41}^{\text{ext}} & T_{42}^{\text{ext}} & T_{43}^{\text{ext}} \\ T_{51}^{\text{ext}} & T_{52}^{\text{ext}} & T_{53}^{\text{ext}} \\ T_{61}^{\text{ext}} & T_{62}^{\text{ext}} & T_{63}^{\text{ext}} \\ — & T_{72}^{\text{ext}} & T_{73}^{\text{ext}} \end{bmatrix} = \begin{bmatrix} 1.4879 & 0.7492 & -0.9107 \\ 1.4837 & 0.0022 & 0.0113 \\ 0.6990 & 0.0155 & 0.0102 \\ — & 0.0000 & -0.8458 \end{bmatrix}. \tag{24}$$

The detailed time scheduling results are shown in Table 3.

5.3. Effect of β on the Scheduling Performance of the LSSC. In the MCLS, responding quickly to customize requirements (including time requirement) is an important objective of time scheduling. Thus, customers' required completion time of a service order may change, and time compression or delay requirement is possible, demanding a certain degree of time flexibility in scheduling from the LSI. In model building, $\beta_i < 0$ means that the service order needs to be finished ahead of time; accordingly, $\beta_i > 0$ means that the service time needs to be delayed. Clearly, the permitted order completion time is directly related to the FLSPs' difficulty in completing the order. This order completion time directly affects the costs of these FLSPs.

In this section, we discuss the influence of the delay (or compression) coefficient of order completion time β on the three objective functions, namely, total satisfaction of FLSPs, service delivery punctuality, and total cost of LSI. Keeping the model parameters unchanged and changing only the value of β, we calculate the corresponding results for Z. These results are shown in Table 4. By plotting the data in Table 4, we obtain Figure 5.

Table 4: The influence of β on comprehensive performance of LSSC.

β	Z
−0.09	0.8482
−0.08	0.8496
−0.05	0.8591
−0.02	0.8603
0	0.9055
0.02	0.9239
0.05	0.9324
0.08	0.9251
0.1	0.9178
0.12	0.9124
0.15	0.9085
0.18	0.8977

Based on Table 4 and Figure 5, the following conclusions can be obtained.

(1) With the increase in β (from negative to positive), Z first increases and then decreases, which means that a reasonable positive tolerance coefficient contributes to achieving the maximal value of comprehensive performance (i.e., when $\beta = 0.05$, comprehensive performance reaches the maximum $Z = 0.93$). Conversely, if β is negative, the maximal value of comprehensive performance cannot be reached. Moreover, a smaller time delay tolerance coefficient (i.e., the service should be operated in time compression) results in poorer comprehensive performance. Therefore, in practice, comprehensive performance

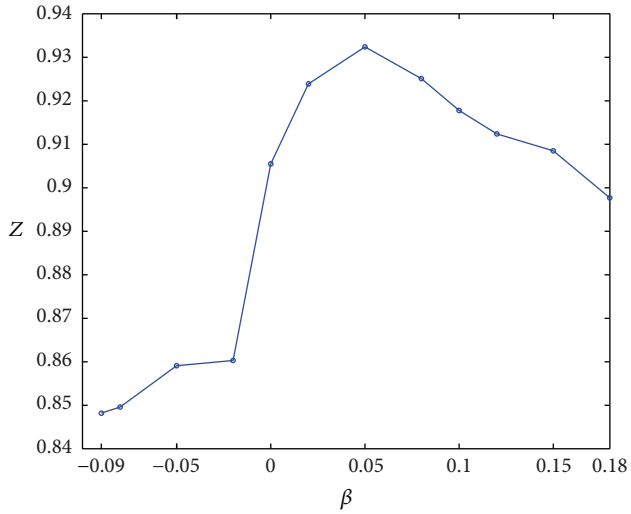


Figure 5: Curve of Z changed with β.

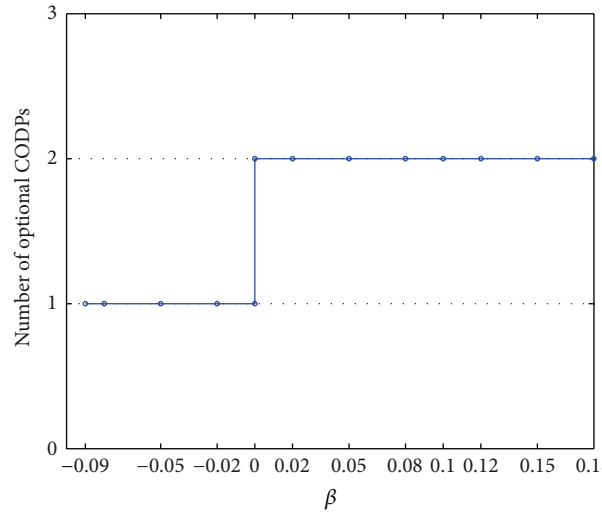


Figure 6: Curve of the number of optional CODPs with β.

can deteriorate when customers request shortening the order completion time of FLSP.

(2) Z significantly increases when β changes from being negative to positive. If the FLSP operates in the case in which the operation time is ahead of schedule and the LSI provides some lag time, then the comprehensive performance of the LSSC significantly improves more than that in the case in which the FLSP operates in a time delay state.

(3) If $\beta < -0.09$, the model has no solution, which means that the LSSC cannot operate in time compression without limit. Furthermore, the LSSC scheduling has certain characteristics, and the order cannot be completed as early as the customer wants it. When a time delay is set by customers, comprehensive performance will not increase all the time, as the increase is followed by a decrease after a certain inflection point. Comprehensive performance will not increase to infinity even when the customer permits a delay in the order completion time. On the contrary, an optimal delay coefficient of order completion time exists.

(4) When β changes from being negative to positive, the range of optional CODP changes from either 4 or 5 to 4 only;that is, the number of optional CODPs is reduced from 2 to 1 (Figure 6). Given the reinforcement to shorten the completion time, the optional CODP range and supply chain flexibility are reduced. In this case, LSI has to sacrifice cost and increase customization to meet the time requirement. Therefore, a reasonable β should be chosen to guarantee an appropriate degree of customization in practice.

5.4. Effect of c on the Scheduling Result of the LSSC. In the model solution, the relationship cost coefficient c is introduced into the solution approach, and the cost objective of the LSI changes to a new constraint. Therefore, pursuing the minimal cost objective is not necessary, but the cost should be kept within a reasonable range. Generally, the relationship cost coefficient c is decided by the LSI, and its size directly influences FLSP satisfaction and LSSC comprehensive performance. In this section, we explore the effect of the relationship cost coefficient of LSI on comprehensive performance.

By changing only the value of c and keeping other parameters unchanged, the corresponding Z and Z^* can be obtained (see Table 5) to explore the relationship between c and LSSC comprehensive performance (denoted by Z and Z^*). In this numerical example, LSSC can operate smoothly when CODP is positioned at $k = 4$ or $k = 5$, but their corresponding comprehensive performances are different. Specifically, the comprehensive performance of $k = 4$ is better than that of $k = 5$.

By plotting the data in Table 5, we obtain Figure 7.

Figure 7 clearly shows that Z increases with the increase in c and ultimately tends to be stable. The comprehensive performance of LSSC increases with the increase in the relationship cost coefficient of the LSI and remains stable after reaching a certain value. When c increases from 0 to 0.05, the slope of the curve is relatively large. Afterwards, the growth in comprehensive performance slows down with the increase in c and ultimately stabilizes at the value 0.9324. The implication is that, after c increases to a certain level, a continued increase in cost will not contribute to the improvement of the comprehensive performance of the LSSC. Moreover, the improvement in supply chain comprehensive performance caused by the increase in the cost relationship coefficient has certain limitations.

5.5. Effect of w on the Scheduling Results. In this paper, $\theta = (1/J_0)\sum_{j=1}^{J_0}(\Delta K_j/K_j)$ denotes the order difference coefficient. w is the order difference tolerance coefficient of LSI, which is decided by LSI. In this section, we explore the influence

Table 5: The influence of c on scheduling result.

c	Z		Z^*
	$k = 4$	$k = 5$	
0.02	$Z = 0.9226$	$Z = 0.8439$	$Z = 0.9226$
0.05	$Z = 0.9297$	$Z = 0.8486$	$Z = 0.9297$
0.10	$Z = 0.9307$	$Z = 0.8511$	$Z = 0.9307$
0.15	$Z = 0.9310$	$Z = 0.8522$	$Z = 0.9310$
0.2	$Z = 0.9324$	$Z = 0.8530$	$Z = 0.9324$
0.25	$Z = 0.9324$	$Z = 0.8536$	$Z = 0.9324$
0.3	$Z = 0.9323$	$Z = 0.8536$	$Z = 0.9323$
0.4	$Z = 0.9324$	$Z = 0.8538$	$Z = 0.9324$

Table 6: Z varied with w.

w	Optimal CODP	Z
0.05	No solution	No solution
0.06	$k = 5$	0.8530
0.1	$k = 5$	0.8530
0.15	$k = 5$	0.8530
0.2	$k = 5$	0.8530
0.25	$k = 4$	0.9324
0.3	$k = 4$	0.9324
0.35	$k = 4$	0.9324
0.4	$k = 4$	0.9324
0.45	$k = 4$	0.9324
0.5	$k = 4$	0.9324
0.55	$k = 4$	0.9324
0.6	$k = 4$	0.9324
0.65	$k = 4$	0.9324

of the LSI's order difference tolerance coefficient w on the scheduling results.

This section discusses the influence of w on the scheduling results by changing only w and keeping $\beta = 0.05$, $\rho = 0.1$, and $c = 0.2$ unchanged. Results in Table 6 were obtained using Matlab 7.8. software.

In Figure 8, the comprehensive performance Z shows a step-shaped growth along with the increase in w. Theoretical analysis of the model shows that the main role of w is to restrict the range of optional CODP. In our numerical examples, the variation in w mainly leads to changes in the selectable range of CODP in set $\{4, 5\}$, considering other constraints. In Figure 9, if w is too small (i.e., $w < 0.06$ in this example), no solution is obtained, indicating that LSSC cannot operate in this case. When $0.06 < w < 0.25$, $k = 5$ leads to optimal comprehensive performance $Z^* = 0.8530$; when $w = 0.25$, $k = 4$ leads to optimal comprehensive performance $Z^* = 0.9324$; when $w > 0.25$, the increase in w no longer increases comprehensive performance. Thus, LSI must set a reasonable value for the order difference tolerance coefficient w. If w is too small (i.e., $w < 0.06$ in this example), the supply chain will not work at all. LSI should make its order difference tolerance coefficient w as large as possible to accumulate more customer orders and obtain mass service effects. At the same time, w should be kept in a proper range, as a too large w is not beneficial to increasing the comprehensive performance of LSSC.

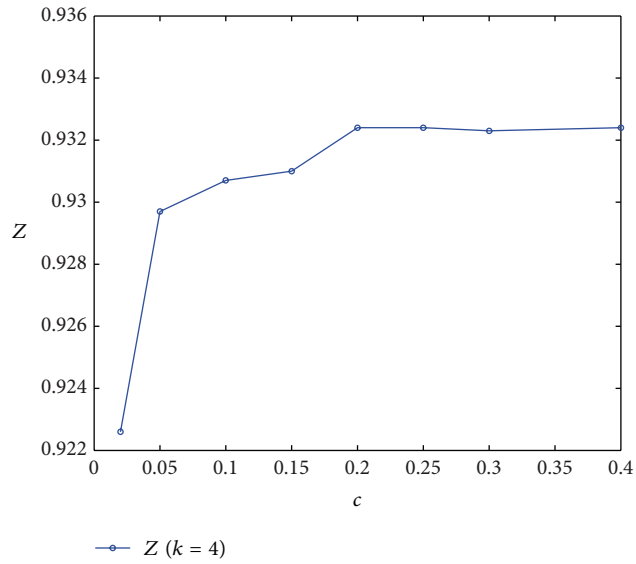


Figure 7: Curve of Z varied with c.

Table 7: Results of Z' varied with ρ.

ρ	Z' when $k = 4$	Z' when $k = 5$	Optimal CODP
0	0.7185	0.7034	$k = 4$
0.05	0.7250	0.7243	$k = 4$
0.10	0.7318	0.7558	$k = 5$
0.15	0.7417	0.7986	$k = 5$
0.2	0.7488	0.8637	$k = 5$

Table 8: The results of Z' varied with ρ ($R = 3$).

ρ	Z' when $k = 4$	Z' when $k = 5$	Optimal CODP
0	0.6359	0.6268	$k = 4$
0.05	0.6461	0.6635	$k = 5$
0.10	0.6595	0.7098	$k = 5$
0.15	0.6734	0.7727	$k = 5$
0.2	0.6885	0.8732	$k = 5$

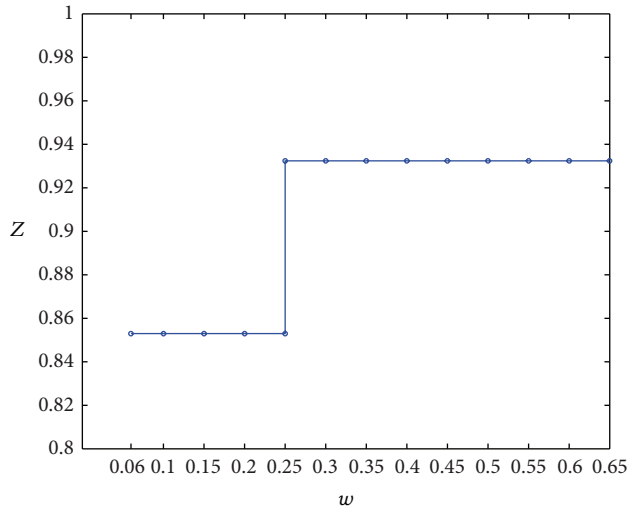


Figure 8: Curve of Z varied with w.

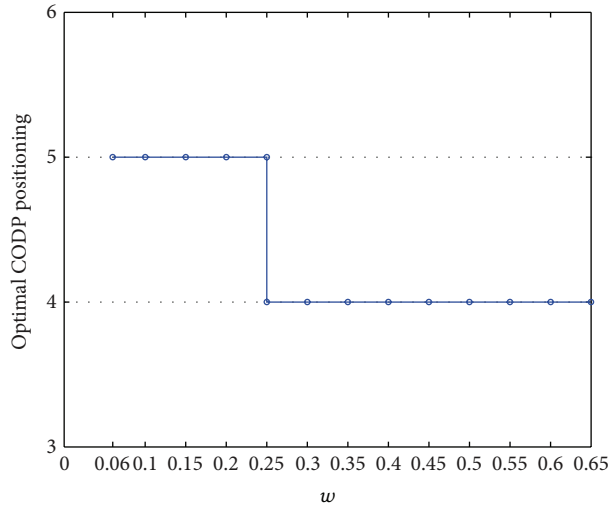


Figure 9: Curve of the optimal CODP varied with w.

5.6. Effect of the Mass Service Effect Coefficient ρ Obtained by the LSI on the Scheduling Results. Success of the MC mode lies in reducing total service cost by realizing the mass service effects in the premise of meeting the customized requirements. In this section, we discuss the influence of the mass service effect coefficient ρ obtained by the LSI on the scheduling results of the LSSC.

The numerical analysis in previous sections mainly focuses on the performance of time scheduling and not much on the cost objective. We regard cost objective as a new constraint. However, to discuss the influence of ρ on the scheduling results in this section, ρ will influence the total cost of the LSSC. Therefore, we consider three subtargets, namely, cost objective, punctual service delivery objective, and satisfaction of all FLSPs, in the overall objective at the same time. The influence of ρ obtained by the LSI is discussed by assigning different weights to these three subtargets.

5.6.1. Effect of the Variation of ρ on the Scheduling Results When the Weights of Z_1, Z_2, and Z_3 are the Same. The same weights are assigned to Z_1, Z_2, and Z_3, that is; their weights are all 1/3. Each of these objective functions should be normalized before synthesis. The minimum value of Z_1 (denoted by $Z_1^{\min}$) should be calculated when the objective functions Z_2 and Z_3 are considered. The resulting overall objective function is shown in

$$\max\ Z' = \frac{1}{3} \times \frac{Z_1^{\min}}{Z_1} + \frac{1}{3} \times (1 - Z_2) + \frac{1}{3} \times Z_3. \quad (25)$$

The other eight constraints, namely (14) to (19) and (22), remain unchanged.

The value of $Z_1^{\min}$ mentioned above is the minimum obtained by changing the selectable range of CODP (i.e., k) and the mass service effect coefficient ρ. Based on Matlab 7.8 calculations, $Z_1^{\min} = 3088.2$ and $k = 5$ at that point. Therefore,

Table 9: The influence of ρ on the scheduling results when R changes.

R	ρ	Z' when $k = 4$, ($R = 3$)	Z' when $k = 5$, ($R = 3$)	Optimal CODP	The threshold of ρ when the optimal CODP changes from $k = 4$ to $k = 5$
3	0	0.6359	0.6268	$k = 4$	0.017
	0.05	0.6461	0.6635	$k = 5$	
	0.10	0.6595	0.7098	$k = 5$	
	0.15	0.6734	0.7727	$k = 5$	
	0.2	0.6885	0.8732	$k = 5$	
2	0	0.6625	0.6530	$k = 4$	0.022
	0.05	0.6722	0.6838	$k = 5$	
	0.10	0.6836	0.7247	$k = 5$	
	0.15	0.6961	0.7814	$k = 5$	
	0.2	0.7090	0.8684	$k = 5$	
1	0	0.7185	0.7034	$k = 4$	0.05
	0.05	0.7250	0.7243	$k = 4$	
	0.10	0.7318	0.7558	$k = 5$	
	0.15	0.7417	0.7986	$k = 5$	
	0.2	0.7488	0.8637	$k = 5$	
1/2	0	0.7738	0.7526	$k = 4$	0.082
	0.05	0.7793	0.7685	$k = 4$	
	0.10	0.7836	0.7894	$k = 5$	
	0.15	0.7900	0.8179	$k = 5$	
	0.2	0.7960	0.8597	$k = 5$	
1/3	0	0.8039	0.7787	$k = 4$	0.115
	0.05	0.8058	0.7890	$k = 4$	
	0.10	0.8093	0.8052	$k = 4$	
	0.15	0.8149	0.8247	$k = 5$	
	0.2	0.8167	0.8588	$k = 5$	

the first four processes are part of the mass service stage, and the remaining ones are part of the customized stage. The specific scheduling results are as follows:

Mass service stage:

$$\begin{aligned}&\left[T_1^{\text{ext}} \quad T_2^{\text{ext}} \quad T_3^{\text{ext}} \quad T_4^{\text{ext}}\right]\\ &\quad = \left[0.6319 \quad -2.6886 \quad -1.1872 \quad -2.0337\right].\end{aligned} \tag{26}$$

Customized stage:

$$\begin{bmatrix} T_{51}^{\text{ext}} & T_{52}^{\text{ext}} & T_{53}^{\text{ext}} \\ T_{61}^{\text{ext}} & T_{62}^{\text{ext}} & T_{63}^{\text{ext}} \\ — & T_{72}^{\text{ext}} & T_{73}^{\text{ext}} \end{bmatrix} = \begin{bmatrix} 1.4994 & -0.0008 & -0.8043 \\ 0.0000 & 0.9926 & 0.0011 \\ — & 0.0000 & -2.2193 \end{bmatrix}. \tag{27}$$

Keeping other parameters such as $\beta = 0.05$ and $w = 0.5$ unchanged, ρ is changed to obtain the corresponding scheduling results using Matlab 7.8 software. Different CODP positions (i.e., different k) produce different scheduling results. The model has a solution only when $k = 4$ and $k = 5$. Specific results are shown in Table 7.

The data in Table 7 are plotted in Figure 10.

Figures 10 and 11 indicate the following.

(1) The comprehensive scheduling performance of LSSC increases with the increase in ρ, no matter where the CODP is positioned.

(2) The two curves in Figure 10 show that when the CODP varies, the comprehensive performance of LSSC also varies. When ρ is relatively small ($\rho < 0.051$), $k = 5$ improves the comprehensive performance of the supply chain. This performance is improved by $k = 4$ when ρ is relatively large ($\rho > 0.051$). The A3 point is the change point for the optimal CODP.

(3) As the mass service effect is the characteristic MCLS, in practice, LSI should choose a reasonable CODP according to ρ. That is, the factor of ρ should be considered in the CODP decision.

5.6.2. Effect of the Variation in ρ on the Scheduling Results When the Weights of Z_1, Z_2, and Z_3, Are Different. Generally, total cost and service delivery punctuality are the most important objectives in scheduling for the LSI. Different LSIs have different attitudes toward the relative importance of cost and service delivery punctuality, thus affecting scheduling results. This section provides an in-depth discussion of the effects of the different objective weights on the scheduling results of the LSSC.

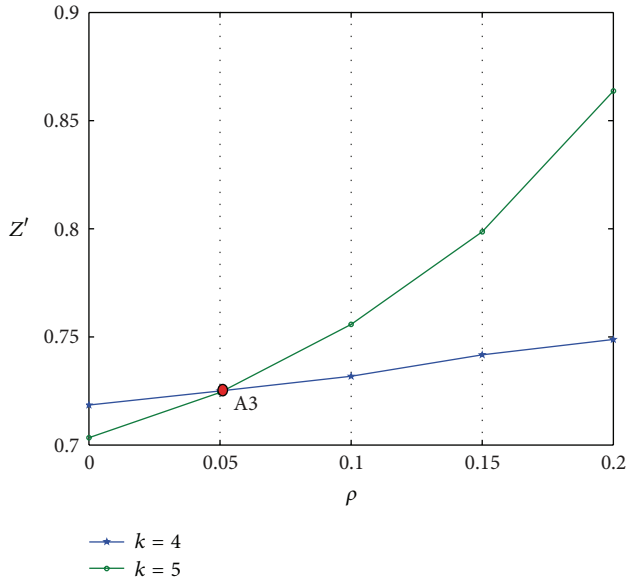


Figure 10: Curve of Z' varied with ρ.

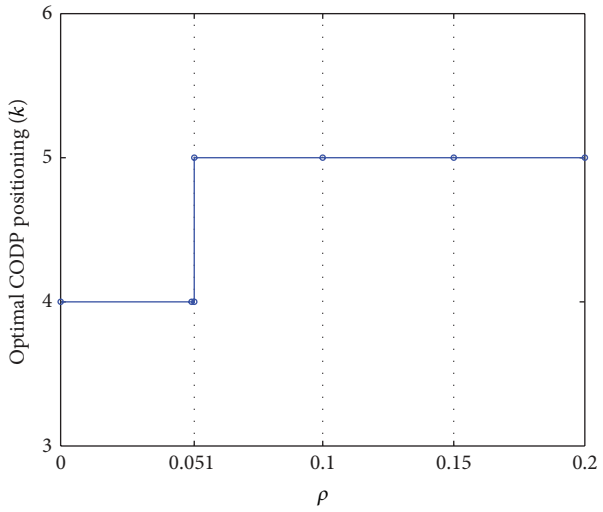


Figure 11: Curve of optimal CODP varied with ρ.

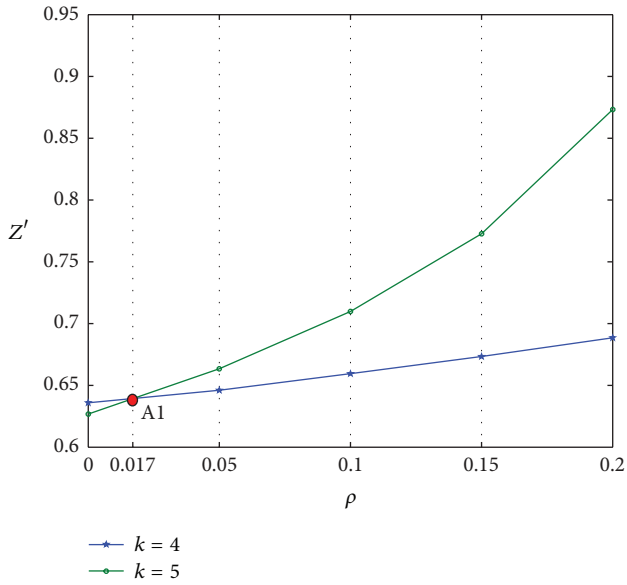


Figure 12: Z' varied with ρ ($R = 3$).

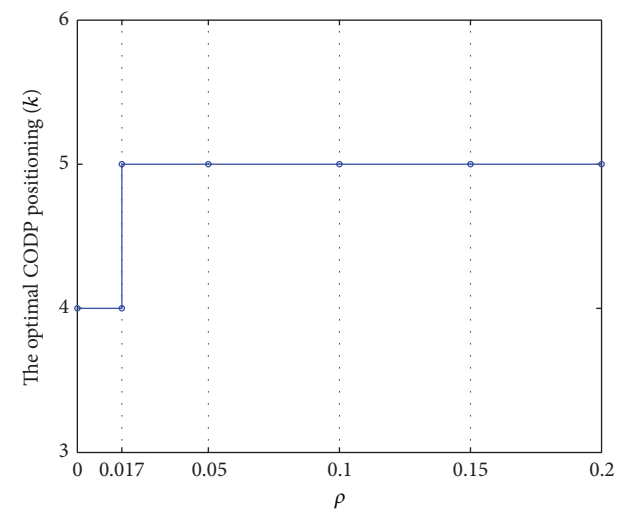


Figure 13: Optimal CODP varied with ρ ($R = 3$).

k_1, k_2, and k_3 refer to the weights of cost objective, punctual service delivery objective, and FLSP satisfaction objective, respectively, in the synthesized objective function. In this section, we introduce the coefficient of weight difference degree R to represent the degree of difference in attitudes toward the relative importance of cost and service delivery punctuality. We let $R = k_1/k_2$. When $R > 1$, the LSI places more focus on cost objective; when $R < 1$, the punctual service delivery objective receives more attention. In the numerical simulation, $k_3 = 1/3$ is unchanged, R is assigned different values, and ρ varies with each value of R. In what follows, we discuss the influence of the variation of ρ on the scheduling results.

(1) Effect of the Variation of ρ on the Scheduling Results When $R = 3$. The results are shown in Table 8.

As shown in Figures 12 and 13, $k = 4$ or 5 is the optimal CODP. The comprehensive performance of scheduling Z' increases with the increase in ρ. A1 is the intersection of the two curves, where $\rho = 0.017$. A1 point is the changing point where the optimal CODP changes from $k = 4$ to $k = 5$.

(2) Effect of the Variation of ρ on the Scheduling Results When R Assumes Other Values. Table 9 shows the influence of ρ on the scheduling results when R assumes other values.

In Table 9, comprehensive performance Z' increases with ρ, regardless of the value of R. The LSSC can choose different locations for the CODP (in this example, either $k = 4$ or $k = 5$). The comprehensive performance of LSSC scheduling differs when the CODP varies. When ρ is relatively small, $k = 4$ is the optimal position. When ρ is relatively large, $k = 5$ is better. For different values of R, a corresponding threshold

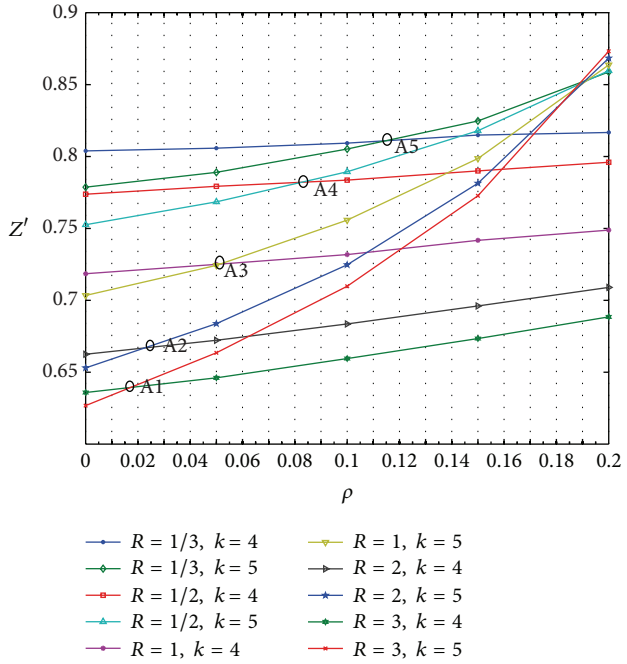


FIGURE 14: Effect of ρ on the scheduling results when R assumes different values.

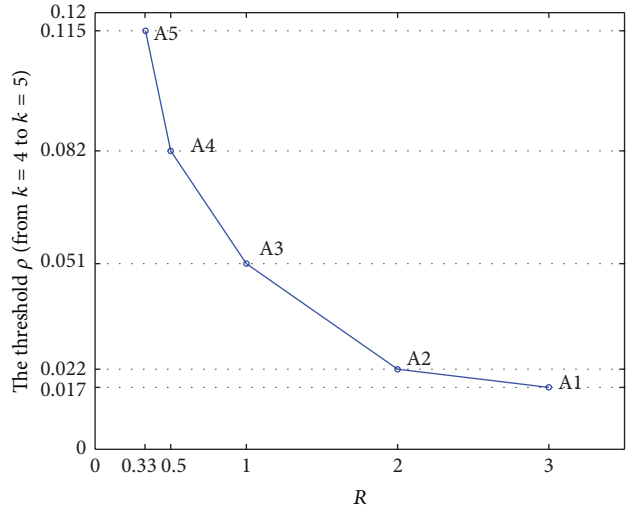


FIGURE 15: Curve of the thresholds of ρ at which the optimal CODP changes from $k = 4$ to $k = 5$.

for ρ exists at which the optimal CODP changes from $k = 4$ to $k = 5$. The data are plotted in Table 9 to better present the variation in these thresholds, as shown in Figure 14.

As shown in Figure 14, the effects of ρ on the scheduling results differ because of the different CODPs and different preferences of the LSI. However, Figure 14 shows that all of these curves tend to increase with ρ. Comparatively, the combined condition of $k = 5$, $R = 3$ makes the slope of this curve the largest, indicating that, in this case, Z' increases the fastest with ρ.

Figure 15 presents the variation thresholds of ρ at which the optimal CODP changes from $k = 4$ to $k = 5$ at different values of R. According to Figure 15, we obtain the following conclusions.

(1) The threshold of ρ shows that the change in the optimal CODP increases with the decline in R. This threshold indicates that the more the LSI pays attention to the cost goal, the smaller the threshold of ρ becomes; the more the LSI pays attention to the punctual service delivery goal, the larger the threshold of ρ becomes. Moreover, the more the LSI pays attention to the cost goal, the greater the motivation to enlarge the CODP position is (from $k = 4$ to $k = 5$), indicating that LSI can change its CODP position when the mass service effect coefficient is relatively small.

(2) When other conditions are the same, customers are better off choosing LSI, whose mass service effect coefficient is relatively large, to obtain a higher level of customization.

(3) Figure 15 shows that the scope of this curve tends to be zero with the increase in R, indicating that the difference in these thresholds of ρ tends to decrease with the increase in R. For example, when R increases from 1/3 to 1/2, the threshold of ρ changes from 0.115 to 0.082, with a difference of 0.033. When R increases from 2 to 3, the threshold of ρ changes from 0.022 to 0.017, with a difference of only 0.005. Thus, for the LSI whose mass service effect coefficient is relatively small, improving R contributes to obtaining scale cost effects. By contrast, for the LSI whose mass service effect coefficient ρ is relatively large, the mass service effects are evident because of the relatively large value of ρ. Therefore, choosing a relatively small value of R can either obtain good mass service effects or improve service delivery punctuality. In this case, a relatively good level of scheduling performance of LSSC can still be reached.

6. Main Conclusions and Management Insights

In this section, we present the main conclusions of this research and explain related insights for researchers. We also discuss management insights for LSI and propose related recommendations for time scheduling decisions.

6.1. Main Conclusions Derived from the Scheduling Model. The following conclusions are based on the previous analysis.

(1) Z first increases and then decreases with the delay coefficient of order completion time β (from negative to positive). The more the order completion time needs to be compressed, the worse the comprehensive performance of the LSSC becomes. When β is reduced to a certain extent, the model has no feasible solution, which means that the

LSSC cannot operate with too much time compression. When a delay in the order completion time is allowed ($\beta > 0$), the increase in β improves comprehensive performance. However, comprehensive performance deteriorates after β increases to a certain extent because of the trade-off relationship between the objective functions Z_2 and Z_3, which cannot be optimal at the same time.

(2) Numerical analysis shows that when the value of β changes from negative to positive, the selectable range of the CODP is reduced from either $k = 4$ or 5 to only $k = 4$, and correspondingly the feasible position of the CODP is reduced from 2 to 1. This reduction indicates that the optimal CODP position tends to move forward with the stronger requirement to complete the customer order ahead of time, thus reducing the mass service processes and increasing customization. Furthermore, the selectable range of the CODP position and the flexibility of the LSSC decrease with the stronger requirement to complete the customer order ahead of time. The time scheduling requirements need to be satisfied by sacrificing cost and increasing customization. Therefore, in practice, a reasonable β must be chosen to guarantee reasonable customization.

(3) Comprehensive performance Z increases and then tends to be stable with the increase in the cost relationship coefficient. Thus, after reaching a certain level, increasing c no longer contributes to improving the comprehensive performance of the LSSC. Improving in supply chain comprehensive performance caused by the increase in the cost relationship coefficient has certain limitations.

(4) Comprehensive performance Z shows step-shaped growth with the increase in the order difference tolerance coefficient w of the LSI. However, after w reaches a certain value, comprehensive performance remains stable. If w is too small (i.e., $w < 0.06$ in this example), the supply chain will not work. Thus, the coefficient w cannot be too small, and LSI needs to enhance its order difference tolerance coefficient w to achieve better scheduling performance.

(5) Whether the weights of Z_1, Z_2, and Z_3 are the same or not, the comprehensive performance of the LSSC improves with the increase in ρ. With same value of ρ, comprehensive performance varies when the CODP locations are different. Therefore, the LSI should choose FLSPs whose mass service effect coefficient is relatively large to obtain more profits.

(6) The preference for the relative concern degree of the LSI for cost and service delivery punctuality leads to differences in the scheduling performance of the LSSC. Using R to represent the ratio of the weights of the cost objective and the punctual service delivery objective, we can find that the threshold of ρ, which reflects changes in the optimal CODP, decreases with the increase in R. Therefore, the more the LSI pays attention to the cost goal, the smaller the threshold of ρ becomes; the more the LSI pays attention to the punctual service delivery goal, the larger the threshold of ρ becomes. When other conditions are the same, customers are better off choosing the LSI whose mass service effect coefficient is relatively large to obtain higher level of customization.

(7) The differences in the thresholds of ρ tend to decrease with the decrease in R. Thus, for the LSI whose mass service effect coefficient is relatively small, improving the weight difference R contributes to the attainment of scale cost effects. Conversely, for the LSI whose mass service effect coefficient is relatively large, reducing the value of R can either obtain good mass service effects or improve service delivery punctuality, ultimately optimizing comprehensive performance.

6.2. Implications for Researchers. This study establishes the LSSC time scheduling model based on the CODP and analyzes the time scheduling problem in the MCLS environment in detail to minimize the total LSSC operation cost, reduce the gap between the expected and the actual time of completing service orders, and maximize the satisfaction of all FLSPs. This study provides theoretical basis for further studies on the scheduling method and performance optimization method of LSSCs in the MCLS environment. For example, we find that both the relationship cost coefficient c and order difference tolerance coefficient w greatly affect the comprehensive performance of the LSSC. The different preferences for the relative concern degree of LSI on cost and service delivery punctuality lead to differences in the scheduling performance of LSSC. The comprehensive performance of the LSSC improves with the increase in the mass service effect coefficient obtained by the LSI. These important conclusions provide some basis for further in-depth studies on time scheduling models. Empirical research can be conducted on the relationship between the comprehensive performance of supply chain scheduling and the factors involved in scheduling. The best decision-making method for the time scheduling process of LSSC can be further discussed. In short, this study provides the necessary theoretical foundation for further development of both theoretical and empirical studies on LSSC scheduling in the MCLS environment.

6.3. Implications for Managers. The conclusions presented in this paper can serve as reference for the participants in LSSC, especially LSI. Results show that through the reasonably designed time scheduling decision and positioning of the optimal CODP, the LSI can maximize comprehensive performance and obtain optimal scheduling results.

We offer several management insights for LSI. First, the LSI can use the strategy of increasing the relationship cost coefficient to improve supply chain performance, but note that this kind of improvement is limited. Second, the LSI must keep a reasonable order difference tolerance coefficient and increase this coefficient to improve performance. Third, regardless of the relative concern degree of LSI for cost and service delivery punctuality, the comprehensive performance of the LSSC improves with the increase in ρ. Therefore, the LSI should choose FLSPs whose mass service effect coefficient is relatively large to obtain better supply chain performance. Fourth, if the mass service effect coefficient obtained by the LSI is relatively small, then the LSI should pay more attention to the cost goal and relax the punctual service delivery goal. Conversely, if the mass service effect coefficient of the FLSP chosen by the LSI is relatively large, then the LSI should pay more attention to the punctual service delivery goal and relax its concerns over the cost goal.

7. Research Limitations and Directions for Future Research

Based on the literature review of the existing scheduling model of the LSSC, this study established the LSSC time scheduling model based on the CODP to minimize the total operational cost for orders in LSSC, minimize the gap between the expected and actual time of completing service orders, and maximize the satisfaction of FLSPs. Numerical analysis was conducted using Matlab 7.8 software. The influence of parameters, such as delay coefficient of order completion time, relationship cost coefficient, and relative concern degree of LSI for cost and service delivery punctuality, on LSSC comprehensive performance was discussed. Through this model, an optimal scheduling plan can be developed. The LSI should set appropriate scheduling parameters to obtain the best scheduling performance of the LSSC.

However, this paper has several limitations. For example, the model solution and analysis are only in accordance with a real numerical example and are thus not representative of all situations in reality. Moreover, the assumption is that the scheduling operation only aims at a set of customer orders and does not consider new arrival orders. In practice, several orders may arrive in succession. Furthermore, our model does not consider the uncertainty of the capacity of FLSPs. In practice, the service-providing capacity of FLSP may be full of uncertainty because of external influence. In the future, a time scheduling model that considers such uncertainty can be established.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant no. 71372156), supported by the Humanity and Social Science Youth foundation of Ministry of Education of China (Grant no. 2013YJC630098), and sponsored by the China State Scholarship Fund and Independent Innovation Foundation of Tianjin University. The suggestions of the reviewers are also gratefully acknowledged.

References

[1] C. Chandra and J. Grabis, "Managing logistics for mass customization: the new production frontier," in *Proceedings of the 6th Biannual World Automation Congress—Image Processing, Biomedicine, Multimedia, Financial Engineering and Manufacturing—International Forum on Multimedia Image Processing (WAC '04)*, pp. 335–340, July 2004.

[2] K. L. Choy, C.-L. Li, S. C. K. So, H. Lau, S. K. Kwok, and D. W. K. Leung, "Managing uncertainty in logistics service supply chain," *International Journal of Risk Assessment and Management*, vol. 7, no. 1, pp. 19–22, 2007.

[3] L. Wei-hua, X. Xue-cai, R. Zheng-xu, and P. Yan, "An emergency order allocation model based on multi-provider in two-echelon logistics service supply chain," *Supply Chain Management*, vol. 16, no. 6, pp. 391–400, 2011.

[4] D. P. van Donk, "Make to stock or make to order: the decoupling point in the food processing industries," *International Journal of Production Economics*, vol. 69, no. 3, pp. 297–306, 2001.

[5] W. Nie and D. L. Kellogg, "How professors of operations management view service operations?" *Production and Operations Management*, vol. 8, no. 3, pp. 339–355, 1999.

[6] W. H. Liu, Y. Yang, X. Li, H. T. Xu, and D. Xie, "A time scheduling model of logistics service supply chain with mass customized logistics service," *Discrete Dynamics in Nature & Society*, Special section, pp. 1–18, 2012.

[7] Y.-C. Wang, "Evaluating flexibility on order quantity and delivery lead time for a supply chain system," *International Journal of Systems Science*, vol. 39, no. 12, pp. 1193–1202, 2008.

[8] N. Mikati, "Dependence of lead time on batch size studied by a system dynamics model," *International Journal of Production Research*, vol. 48, no. 18, pp. 5523–5532, 2010.

[9] A. A. Taleizadeh, S. T. A. Niaki, N. Shafii, R. G. Meibodi, and A. Jabbarzadeh, "A particle swarm optimization approach for constraint joint single buyer-single vendor inventory problem with changeable lead time and (r,Q) policy in supply chain," *International Journal of Advanced Manufacturing Technology*, vol. 51, no. 9–12, pp. 1209–1223, 2010.

[10] B. J. Pine II and S. Davis, *Mass Customization: The New Frontier in Business Competition*, Harvard Business School Press, Boston, Mass, USA, 1993.

[11] G. da Silveira, D. Borenstein, and F. S. Fogliatto, "Mass customization: literature review and research directions," *International Journal of Production Economics*, vol. 72, no. 1, pp. 1–13, 2001.

[12] F. S. Fogliatto, G. J. C. da Silveira, and D. Borenstein, "The mass customization decade: an updated review of the literature," *International Journal of Production Economics*, vol. 138, no. 1, pp. 14–25, 2012.

[13] Y. X. Feng, B. Zheng, J. R. Tan, and Z. Wei, "An exploratory study of the general requirement representation model for product configuration in mass customization mode," *International Journal of Advanced Manufacturing Technology*, vol. 40, no. 7-8, pp. 785–796, 2009.

[14] D. Yang, M. Dong, and X. X. Chang, "A dynamic constraint satisfaction approach for configuring structural products under mass customization," *Engineering Applications of Artificial Intelligence*, no. 25, pp. 1723–1737, 2012.

[15] C. Welborn, "Customization index: evaluating the flexibility of operations in a mass customization environment," *The Icfai University Journal of Operation Management*, no. 8, pp. 6–15, 2009.

[16] X. F. Shao, "Integrated product and channel decision in mass customization," *IEEE Transactions on Engineering Management*, no. 60, pp. 30–45, 2013.

[17] H. Wang, "Defects tracking in mass customisation production using defects tracking matrix combined with principal component analysis," *Internationlal Journal of Production Research*, no. 51, pp. 1852–1868, 2013.

[18] D.-C. Li, F. M. Chang, and S.-C. Chang, "The relationship between affecting factors and mass-customisation level: the case of a pigment company in Taiwan," *International Journal of Production Research*, vol. 48, no. 18, pp. 5385–5395, 2010.

[19] B. Yang, N. D. Burns, and C. J. Backhouse, "Management of uncertainty through postponement," *International Journal of Production Research*, vol. 42, no. 6, pp. 1049–1064, 2004.

[20] R. I. van Hoek, "The rediscovery of postponement a literature review and directions for research," *Journal of Operations Management*, vol. 19, no. 2, pp. 161–184, 2001.

[21] W. Yu and L. Jie, "Supply chain model based on multi-CODP in mass dynamic customization," in *Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII '08)*, pp. 252–255, December 2008.

[22] F. Wang, "The mass customization system based on multi-CODPs," in *Proceedings of the Chinese Control and Decision Conference (CCDC '11)*, pp. 4261–4265, May 2011.

[23] H. Philip and H. C. G. Amos, "Dynamic implications of customer order decoupling point positioning," *Journal of Manufacturing Technology Management*, vol. 22, no. 8, pp. 1032–1042, 2011.

[24] W. Alderson, "Marketing efficiency and the principle of postponement," *Cost and Profit Outlook*, vol. 3, no. 4, pp. 15–18, 1950.

[25] R. D. Shapiroe, "Get leverage from logistics," *Harvard Business Review*, no. 62, pp. 119–126, 1950.

[26] D. J. Bowerson and D. J. Closs, *Logistican Management, The Integrated Supply Chain Process*, McGraw-Hill, New York, NY, USA, 1996.

[27] J. L. Larry, A. H. James, H. Jerry, and W. G. Gregory, "Modeling uncertain forecast accuracy in supply chains with postponement," *Journal of Business Logistics*, vol. 30, no. 1, pp. 19–31, 2009.

[28] G. Jury and K. Matteo, "Product and process modularity: improving flexibility and reducing supplier failure risk," *International Journal of Production Research*, vol. 51, no. 19, pp. 5757–5770, 2013.

[29] B. Yang and Y. Yang, "Postponement in supply chain risk management: a complexity perspective," *International Journal of Production Research*, vol. 48, no. 7, pp. 1901–1912, 2010.

[30] D. Tang and J. Chen, "Identification of postponement point in service delivery process: a description model," in *Proceedings of the 6th International Conference on Service Systems and Service Management (ICSSSM '09)*, pp. 335–339, June 2009.

[31] N. G. Hall and C. N. Potts, "Supply chain scheduling: batching and delivery," *Operations Research*, vol. 51, no. 4, pp. 566–674, 2003.

[32] C.-Y. Lee, S. Piramuthu, and Y.-K. Tsai, "Job shop scheduling with a genetic algorithm and machine learning," *International Journal of Production Research*, vol. 35, no. 4, pp. 1171–1191, 1997.

[33] P. R. Philipoom, "The choice of dispatching rules in a shop using internally set due-dates with quoted leadtime and tardiness costs," *International Journal of Production Research*, vol. 38, no. 7, pp. 1641–1655, 2000.

[34] Z.-L. Chen and N. G. Hall, "Supply chain scheduling: conflict and cooperation in assembly systems," *Operations Research*, vol. 55, no. 6, pp. 1072–1089, 2007.

[35] X. G. Xu and X. Y. Li, "Customer order decoupling point selection model in mass customization based on MAS," in *Proceedings of the 1st International Symposium on Digital Manufacture*, pp. 677–681, 2006.

[36] J. M. Yao, "Scheduling optimisation of co-operator selection and task allocation in mass customisation supply chain based on collaborative benefits and risks," *International Journal of Production Research*, no. 51, pp. 2219–2239, 2013.

[37] J. Yao and L. Liu, "Optimization analysis of supply chain scheduling in mass customization," *International Journal of Production Economics*, vol. 117, no. 1, pp. 197–211, 2009.

[38] M. Dawande, H. N. Geismar, N. G. Hall, and C. Sriskandarajah, "Supply chain scheduling: distribution systems," *Production and Operations Management*, vol. 15, no. 2, pp. 243–261, 2006.

[39] N. Mishra, A. K. Choudhary, and M. K. Tiwari, "Modeling the planning and scheduling across the outsourcing supply chain: a Chaos-based fast Tabu-SA approach," *International Journal of Production Research*, vol. 46, no. 13, pp. 3683–3715, 2008.

[40] E. Selvarajah and G. Steiner, "Approximation algorithms for the supplier's supply chain scheduling problem to minimize delivery and inventory holding costs," *Operations Research*, vol. 57, no. 2, pp. 426–438, 2009.

[41] R. Bhatnagar, P. Mehta, and C. Chong Teo, "Coordination of planning and scheduling decisions in global supply chains with dual supply modes," *International Journal of Production Economics*, vol. 131, no. 2, pp. 473–482, 2011.

[42] W.-H. Liu, J.-H. Ji, and L. Zhou, "An order allocation model in two-echelon logistics service supply chain," *Journal of Shanghai Jiaotong University*, vol. 42, no. 9, pp. 1524–1533, 2008.

[43] W. H. Liu, C. L. Liu, and M. Y. Ge, "An order allocation model for the two-echelon logistics service supply chain based on cumulative prospect theory," *Journal of Purchasing and Supply Management*, no. 19, pp. 39–48, 2013.

[44] U. Tosun, T. Dokeroglu, and A. Cosar, "A robust island parallel genetic algorithm for the quadratic assignment problem," *International Journal of Production Research*, vol. 51, no. 14, pp. 4117–4133, 2013.

[45] Z. Nikdel and H. Beigy, "A genetic programming-based learning algorithms for pruning cost-sensitive classifiers," *International Journal of Computational Intelligence and Applications*, vol. 11, no. 2, pp. 1–16, 2012.

[46] A. Mitra and D. Kundu, "Genetic algorithms based robust frequency estimation of sinusoidal signals with stationary errors," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 3, pp. 321–330, 2010.

[47] H. Suzuki, H. Sawai, and W. Piaseczny, "Chemical genetic algorithms—evolutionary optimization of binary-to-real-value translation in genetic algorithms," *Artificial Life*, vol. 12, no. 1, pp. 89–115, 2006.

[48] Z. Jia, X. Lu, J. Yang, and D. Jia, "Research on job-shop scheduling problem based on genetic algorithm," *International Journal of Production Research*, vol. 49, no. 12, pp. 3585–3604, 2011.

[49] G. Meja, C. Montoya, J. Cardona, and A. L. Castro, "Petri nets and genetic algorithms for complex manufacturing systems scheduling," *International Journal of Production Research*, vol. 50, no. 3, pp. 791–803, 2012.

[50] C. Y. Cheng, T. L. Chen, L. C. Wang, and Y. Y. Chen, "A genetic algorithm for the multi-stage and parallel-machine scheduling problem with job splitting—a case study for the solar cell industry," *International Journal of Production Research*, vol. 51, no. 16, pp. 4755–4777, 2013.

[51] D. L. Giovanni, G. Massi, and F. Pezzella, "An adaptive genetic algorithm for large-size open stack problems," *International Journal of Production Research*, vol. 51, no. 3, pp. 682–697, 2013.

[52] H.-H. Yang and S.-W. Wang, "Solving the 0/1 knapsack problem using rough sets and genetic algorithms," *Journal of the Chinese Institute of Industrial Engineers*, vol. 28, no. 5, pp. 360–369, 2011.

[53] M. Sarabian and L. S. Lee, "A modified partially mapped multicrossover genetic algorithm for two-dimensional bin packing problem," *Journal of Mathematics and Statistics*, vol. 6, no. 2, pp. 157–162, 2010.

[54] S. Küçükpetek, F. Polat, O. gcaron, and H. uztüzün, "Multilevel graph partitioning: an evolutionary approach," *Journal of the Operational Research Society*, vol. 56, no. 5, pp. 549–562, 2005.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 547573, 8 pages
http://dx.doi.org/10.1155/2014/547573

Research Article

Best Possible Approximation Algorithms for Single Machine Scheduling with Increasing Linear Maintenance Durations

Xuefei Shi and Dehua Xu

School of Science, East China Institute of Technology, Nanchang, Jiangxi 330013, China

Correspondence should be addressed to Dehua Xu; dhxu@ecit.cn

Received 6 November 2013; Accepted 19 December 2013; Published 13 February 2014

Academic Editors: J. G. Barbosa and D. Oron

Copyright © 2014 X. Shi and D. Xu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We consider a single machine scheduling problem with multiple maintenance activities, where the maintenance duration function is of the linear form $f(t) = a+bt$ with $a \geq 0$ and $b > 1$. We propose an approximation algorithm named FFD-LS2I with a worst-case bound of 2 for problem. We also show that there is no polynomial time approximation algorithm with a worst-case bound less than 2 for the problem with $b \geq 0$ unless $P = \text{NP}$, which implies that the FFD-LS2I algorithm is the best possible algorithm for the case $b > 1$ and that the FFD-LS algorithm, which is proposed in the literature, is the best possible algorithm for the case $b \leq 1$ both from the worst-case bound point of view.

1. Introduction

Scheduling with machine maintenance has attracted many researchers' attention in the last two decades (see, e.g., [1–13]), where high performance approximation algorithms are of great value not only theoretically but also practically.

In a recent paper, Xu et al. [14] study a single machine scheduling problem with multiple maintenance activities which can be formally described as follows. There are n independent jobs $J_1, J_2, \ldots, J_n$ to be processed on a single machine. The processing time of job J_i is p_i. All jobs are nonpreemptive and are available at time zero. The machine can process at most one job at a time. Assume that the machine has just finished a dummy maintenance activity at time zero. The length of the time interval between any two consecutive maintenance activities is T and prefixed. The amount of time needed to perform one maintenance activity is an increasing linear function $T_M(t) = a + bt$ of the total processing time t of the jobs that are processed after its latest previous maintenance activity, where a and b are prefixed nonnegative real numbers with $b \leq 1$. The objective is to schedule all the jobs to the machine such that the makespan, that is, the completion time of the last finished job, is minimized. Extending the well-known three-field $\alpha|\beta|\gamma$ classification scheme suggested by Graham et al. [15], the problem is denoted by $1, MS[T, T], T_M(t) = a + bt, b \leq 1 \parallel C_{\max}$ in Xu et al. [14]. For simplicity, in what follows, we denote this problem by $P_{(b\leq 1)}$ and its counterpart where $b > 1$ by $P_{(b>1)}$. Additionally, the problem is denoted by $P_{(b>0)}$ if $b > 0$.

Xu et al. [14] first propose an approximation algorithm named FFD-LS for the problem $P_{(b\leq 1)}$ and then show that its worst-case bound is 2. The underlying idea of their FFD-LS algorithm is straightforward. Viewing the jobs as items, it first packs these items into some bins with the same capacity of T by the classical bin-packing algorithm FFD. Let the "jobs" in each used bin plus a maintenance activity whose duration is computed according to the maintenance function $T_M(t)$ be viewed as a hybrid job. Assign all the hybrid jobs except the last one (i.e., the one with the largest index) to the machine by the classical LS algorithm. Assign the last hybrid jobs to the machine.

For the sake of convenience and completeness, the bin-packing problem, the FFD algorithm, the LS algorithm, and the FFD-LS algorithm are presented as follows.

Bin-Packing Problem (see, e.g., Coffman et al. [16]). Given n items $a_1, a_2, \ldots, a_n$, each with a size $s(a_i) \in (0, W]$, we are asked to pack them into a minimum number of

W-capacity bins (i.e., partition them into a minimum number m of subsets $B_1, B_2, \ldots, B_m$ such that $\sum_{a_i \in B_j} s(a_i) \le W, 1 \le j \le m$).

Algorithm FFD (see, e.g., Coffman et al. [16]). Sort all the items such that $s(a_1) \ge s(a_2) \ge \cdots \ge s(a_n)$; for $i = 1, 2, \ldots, n$, item a_i is packed in the first (lowest-indexed) bin into which it will fit; that is, if there is any partially filled bin B_j with $\text{level}(B_j) \le W - s(a_i)$ where $\text{level}(B_j)$ is the sum of the sizes of the items that have been packed into bin B_j, we place a_i in the lowest-indexed bin having this property. Otherwise, we start a new bin with a_i as its first item.

Algorithm LS (see, e.g., Pinedo [17]). Put all the jobs on a list in arbitrary order; then process the jobs consecutively as early as possible.

Algorithm FFD-LS (see Xu et al. [14])

Step 1. Construct a bin-packing instance as follows. There are n items $a_1, a_2, \ldots, a_n$; the size of item a_i is p_i, and the capacity of each bin is T. Using the FFD algorithm, assume that we obtain k used bins $B_1, B_2, \ldots, B_k$. Let bin B_i be denoted as $(a_1^{(i)}, a_2^{(i)}, \ldots, a_{k_i}^{(i)})$, where k_i is the number of items in B_i and $a_j^{(i)}$ is the jth item assigned to B_i.

Step 2. For $i = 1, 2, \ldots, k$, let $\mathcal{J}_i = (J_1^{(i)}, J_2^{(i)}, \ldots, J_{k_i}^{(i)}, \emptyset_i, M^{(i)})$, where $J_j^{(i)}$ is the job with a processing time of $p_j^{(i)}$ corresponding to item $a_j^{(i)}$, $\emptyset_i$ is a dummy job with the processing time of $T - \sum_{j=1}^{k_i} p_j^{(i)}$, and $M^{(i)}$ denotes a maintenance activity with the length of $a + \text{level}(B_i)b$.

Step 3. Let $\mathcal{J}_i$ be viewed as a single job with the processing time of $T + a + \text{level}(B_i)b$. Assign $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{k-1}$ to the machine by the LS algorithm. Assign $\mathcal{J}_k$ to the machine.

Although the FFD-LS algorithm is originally proposed for $P_{(b\le 1)}$, it can also be applied to $P_{(b>1)}$ without any modification. However, as can be seen in the next section, there exists an example showing that the worst-case bound of the FFD-LS algorithm can be arbitrarily large for $P_{(b>1)}$. As a result, we then focus our attention on studying its worst-case bound for $P_{(b>1)}$ in detail. Specifically, we think of each interval between two consecutive maintenance activities as a batch with a capacity T and show that its worst-case bound is 2 for $P_{(b>1)}$ if the number of nonempty batches in the FFD-LS schedule is not 2. In order to avoid the arbitrarily large worst-case bound, in Section 3, we modify the FFD-LS algorithm and get a new approximation algorithm named FFD-LS2I algorithm with a worst-case bound of 2 for $P_{(b>1)}$ by adding a step to the FFD-LS algorithm. In Section 4, we study the nonapproximability of the problem and show that there is no polynomial time approximation algorithm with a worst-case bound less than 2 for the problem $P_{(b>0)}$, which implies that the FFD-LS2I algorithm is the best possible algorithm for $P_{(b>1)}$ and that the FFD-LS is the best possible algorithm for $P_{(b\le 1)}$ both from the worst-case bound point of view. Finally, in Section 5, we give an answer to another open problem proposed in Xu et al. [14].

2. Worst-Case Bound of the FFD-LS Algorithm for $P_{(b>1)}$

Example 1. Consider the scheduling problem $P_{(b>1)}$ with the following instance: $n = 2, T = b$, $p_1 = b$, and $p_2 = 1$. It is easy to see that the FFD-LS algorithm assigns job J_1 to the first batch and J_2 to the second batch and that the corresponding makespan is $C_{\max}^{\text{FFD-LS}} = b + a + b^2 + 1 = b^2 + b + a + 1$. Note that the optimal schedule assigns job J_2 to the first batch and J_1 to the second batch and that the optimal makespan is $C_{\max}^* = 3b + a$. Clearly, $C_{\max}^{\text{FFD-LS}}/C_{\max}^* \to \infty$ as $b \to \infty$. In other words, the worst-case bound of the FFD-LS algorithm can be arbitrarily large.

Now, let $\Lambda = \sum_{i=1}^{n} p_i$. Let S^* be an optimal schedule with k^* nonempty batches $\mathcal{B}_1^*, \ldots, \mathcal{B}_{k^*}^*$ which are indexed according to the processing order. Recall that we assume that there are totally k bins being used by the FFD algorithm in the first step of the FFD-LS algorithm. So, for consistency, we may assume that there are totally k nonempty batches in the FFD-LS schedule. For convenience, in what follows, these batches are denoted by $\mathcal{B}_1, \ldots, \mathcal{B}_k$ which are also indexed according to the processing order. Let the total processing times of the jobs in batch $\mathcal{B}_i^*$ and batch $\mathcal{B}_i$ be denoted by λ_i^* and λ_i, respectively. It is easy to see that the makespan of the optimal schedule is

$$C_{\max}^* = (k^* - 1)(T + a) + b(\Lambda - \lambda_{k^*}^*) + \lambda_{k^*}^* \quad (1)$$

and that the makespan of the FFD-LS schedule is

$$C_{\max}^{\text{FFD-LS}} = (k - 1)(T + a) + b(\Lambda - \lambda_k) + \lambda_k. \quad (2)$$

In what follows, we shall study the worst-case bound of the FFD-LS algorithm for $P_{(b>1)}$ in detail, which can help us develop a more sophisticated approximation algorithm. Before the study of the worst-case bound, we first present some lemmas.

Lemma 2. *For the problem $P_{(b>1)}$, the total processing time of the jobs in any two nonempty batches is strictly larger than T in S^*.*

Proof (by contradiction). It suffices to show that $\lambda_i^* + \lambda_j^* > T$ for $1 \le i < j \le k^*$.

If there exist two batches, say $\mathcal{B}_i^*$ and $\mathcal{B}_j^*$, such that $\lambda_i^* + \lambda_j^* \le T$, where $1 \le i < j \le k^*$, consider the following two cases.

Case 1 ($1 \le i < j < k^*$). Note that $\lambda_i^* + \lambda_j^* \le T$. Now, reschedule all the jobs of $\mathcal{B}_i^*$ into $\mathcal{B}_j^*$ and delete $\mathcal{B}_i^*$ with the corresponding maintenance activity. Let the new schedule be denoted by S'. It is easy to see that S' is a feasible schedule with a makespan of

$$C'_{\max} = (k^* - 2)(T + a) + b(\Lambda - \lambda_{k^*}^*) + \lambda_{k^*}^*. \quad (3)$$

Comparing (1) with (3), we have $C'_{\max} < C_{\max}^*$, which contradicts the optimality of S^*.

Case 2 ($1 \le i < k^*$ and $j = k^*$). Similar to Case 1, reschedule all the jobs of $\mathcal{B}_i^*$ into $\mathcal{B}_j^*$ and delete $\mathcal{B}_i^*$ with

its corresponding maintenance. Let the new schedule be denoted by S''. It is easy to see that S'' is a feasible schedule with a makespan of

$$C''_{\max} = (k^* - 2)(T + a) + b(\Lambda - \lambda^*_{k^*} - \lambda^*_i) + (\lambda^*_{k^*} + \lambda^*_i). \tag{4}$$

Recall that $b > 1$. Comparing (1) with (4), we have $C''_{\max} < C^*_{\max}$, which again contradicts the optimality of S^*.

This completes the proof. □

Lemma 3. *For problem $P_{(b>1)}$, let S' be a feasible schedule with two nonempty batches and let S'' be a feasible schedule with three nonempty batches. Let the total processing times of the jobs in the first batch of S' and in the second batch of S' be denoted by λ'_1 and λ'_2, respectively. Let the total processing times of the jobs in the first batch of S'', in the second batch of S'', and in the third batch of S'' be denoted by λ''_1, λ''_2 and λ''_3, respectively. If $\lambda'_1 + \lambda'_2 > T$ and $\lambda''_i + \lambda''_j > T$ for $1 \le i < j \le 3$, then one has $C'_{\max} < C''_{\max}$, where $C'_{\max}$ and $C''_{\max}$ are the makespans of S' and S'', respectively.*

Proof. It is easy to see that

$$C'_{\max} = T + a + b\lambda'_1 + \lambda'_2, \tag{5}$$

$$C''_{\max} = 2(T + a) + b(\lambda''_1 + \lambda''_2) + \lambda''_3. \tag{6}$$

Combining (5) with (6), we have

$$C'_{\max} = C''_{\max} + (\lambda'_2 - T) - a + b(\lambda'_1 - \lambda''_1 - \lambda''_2) - \lambda''_3. \tag{7}$$

Recall that $\lambda''_1 + \lambda''_2 > T$. So we have

$$C'_{\max} < C''_{\max} + (\lambda'_2 - T) - a + b(\lambda'_1 - T) - \lambda''_3. \tag{8}$$

Note that $\lambda'_1 \le T$ and $\lambda'_2 \le T$. So we have

$$C'_{\max} < C''_{\max}. \tag{9}$$

This completes the proof. □

Lemma 4. $k \le 3k^*/2$.

Proof. View each interval between two consecutive maintenance activities as a bin with capacity T and the jobs J_i as items a_i with a size of p_i. Let k' be the minimum number of nonempty bins that is needed to pack all the n items. It is well known that $k \le 3k'/2$ (see Simchi-Levi [18]). Note that $k' \le k^*$. So we have $k \le 3k^*/2$. This completes the proof. □

Lemma 5. *For the problem $P_{(b>1)}$, if $T < \Lambda \le 2T$, then $2 \le k^* \le 3$ and $2 \le k \le 3$.*

Proof. By $T < \Lambda$, it is easy to see that $2 \le k^*$ and $2 \le k$ since that one batch is clearly not sufficient in any feasible schedule. On the other hand, by Lemma 2, we have $k^* \le 3$ for otherwise we have $\Lambda > 2T$, a contradiction. Note that the total processing time of the jobs in any two batches in the FFD-LS schedule is strictly larger than T. We thus have $k \le 3$. To sum up, we have $2 \le k^* \le 3$ and $2 \le k \le 3$. This completes the proof. □

Now, we are ready to evaluate the worst-case bound of the FFD-LS algorithm in detail.

Theorem 6. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS algorithm is not greater than 2 if $\Lambda > 2T$.*

Proof. By (1), we have

$$k^* = 1 + \frac{C^*_{\max} + (b-1)\lambda^*_{k^*} - b\Lambda}{T + a}. \tag{10}$$

Note that $k \le 3k^*/2$ (see Lemma 4). So we have

$$k \le \frac{3}{2}\left(1 + \frac{C^*_{\max} + (b-1)\lambda^*_{k^*} - b\Lambda}{T + a}\right). \tag{11}$$

Substituting (11) into (2), we have

$$C^{\text{FFD-LS}}_{\max} \le \frac{3}{2}C^*_{\max} + \frac{1}{2}(T + a) + \frac{3}{2}(b-1)\lambda^*_{k^*} - (b-1)\lambda_k - \frac{b}{2}\Lambda. \tag{12}$$

Now, consider the following two cases.

Case 1 ($\Lambda \ge 3T$). In this case, we have

$$\frac{b}{2}\Lambda \ge \frac{b}{2}3T > \frac{3}{2}(b-1)T. \tag{13}$$

Substituting (13) into (12), we have

$$C^{\text{FFD-LS}}_{\max} \le \frac{3}{2}C^*_{\max} + \frac{1}{2}(T + a) + \frac{3}{2}(b-1)(\lambda^*_{k^*} - T) - (b-1)\lambda_k. \tag{14}$$

Note that $\lambda^*_{k^*} \le T$, $-(b-1)\lambda_k \le 0$, and $T + a < C^*_{\max}$. Combining these three inequalities with (14), we have $C^{\text{FFD-LS}}_{\max} < 2C^*_{\max}$ and we are done.

Case 2 ($2T < \Lambda < 3T$). In this case, it is easy to see that $k^* \ge 3$ because two batches are obviously not sufficient in any optimal solution. Note that the total processing time of the jobs in any two nonempty batches is strictly larger than T in the FFD-LS schedule. We thus have $k \le 5$ for otherwise the total processing times of the jobs will be strictly larger than $3T$, which violates the assumption of this case.

Combining $k^* \ge 3$ with (1), we have

$$C^*_{\max} \ge 2(T + a) + b(\Lambda - \lambda^*_{k^*}) + \lambda^*_{k^*}. \tag{15}$$

By simple algebra, we have

$$C^*_{\max} \ge 2(T + a) + b\Lambda - (b-1)\lambda^*_{k^*}. \tag{16}$$

Note that $0 < \lambda^*_{k^*} \le T$. So we have

$$C^*_{\max} \ge 2(T + a) + b\Lambda - (b-1)T. \tag{17}$$

Similarly, combining $k \le 5$, $0 < \lambda_k \le T$, and (2), we have

$$C_{\max}^{\text{FFD-LS}} < 4(T + a) + b\Lambda. \tag{18}$$

By simple algebra, (18) becomes

$$C_{\max}^{\text{FFD-LS}} < 2(2(T + a) + b\Lambda - (b - 1)T) - b(\Lambda - 2T) - 2T. \tag{19}$$

Substituting (17) into (19), we have

$$C_{\max}^{\text{FFD-LS}} < 2C_{\max}^{*} - b(\Lambda - 2T) - 2T. \tag{20}$$

Recall that $2T < \Lambda$; we thus have $C_{\max}^{\text{FFD-LS}} < 2C_{\max}^{*}$.

This completes the proof. □

Theorem 7. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS algorithm is not greater than 2 if $T < \Lambda \le 2T$ and $k = 3$.*

Proof. Substituting $k = 3$ into (2), we have

$$C_{\max}^{\text{FFD-LS}} = 2(T + a) + b\Lambda - (b - 1)\lambda_3. \tag{21}$$

By Lemma 5, we have $2 \le k^{*} \le 3$. Now, consider the following two cases.

Case 1 ($k^{*} = 2$). Substituting $k^{*} = 2$ into (1), we have

$$C_{\max}^{*} \ge T + a + b(\Lambda - \lambda_2^{*}) + \lambda_2^{*}. \tag{22}$$

Combining (22) with (21), we have

$$C_{\max}^{\text{FFD-LS}} = 2C_{\max}^{*} - (b - 1)(\Lambda + \lambda_3 - 2\lambda_2^{*}) - \Lambda. \tag{23}$$

Note that $\Lambda = \lambda_1 + \lambda_2 + \lambda_3$. Combining this with (23), we have

$$C_{\max}^{\text{FFD-LS}} \le 2C_{\max}^{*} - (b - 1)((\lambda_1 + \lambda_3 - \lambda_2^{*}) + (\lambda_2 + \lambda_3 - \lambda_2^{*})) - \Lambda. \tag{24}$$

Note that $\lambda_1 + \lambda_3 > T \ge \lambda_2^{*}$ and that $\lambda_2 + \lambda_3 > T \ge \lambda_2^{*}$. So, by (24), we have $C_{\max}^{\text{FFD-LS}} \le 2C_{\max}^{*}$.

Case 2 ($k^{*} = 3$). Substituting $k^{*} = 3$ into (1), we have

$$C_{\max}^{*} = T + a + b(\Lambda - \lambda_3^{*}) + \lambda_3^{*}. \tag{25}$$

Combining (25) with (21), we have

$$C_{\max}^{\text{FFD-LS}} = C_{\max}^{*} + (b - 1)(\lambda_3^{*} - \lambda_3). \tag{26}$$

Note that $\lambda_3^{*} - \lambda_3 \le T$ and $b - 1 \le b$. So we have

$$C_{\max}^{\text{FFD-LS}} \le C_{\max}^{*} + bT. \tag{27}$$

Note that $\Lambda - \lambda_3^{*} = \lambda_1^{*} + \lambda_2^{*} > T$. Combining this with (25), we have $C_{\max}^{*} > bT$. Substituting this into (27), we have $C_{\max}^{\text{FFD-LS}} < 2C_{\max}^{*}$.

This completes the proof. □

Theorem 8. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS algorithm is not greater than 2 if $k \ne 2$.*

Proof. Consider the following three cases.

Case 1 ($k = 1$). It is trivially obvious that in this case we have $C_{\max}^{\text{FFD-LS}} = C_{\max}^{*}$, and we are done.

Case 2 ($k = 3$). Clearly, in this case, we have $\Lambda > T$. Now, if $T < \Lambda \le 2T$, then, by Theorem 7, we know that the worst-case bound of the FFD-LS algorithm is not greater than 2, and we are done. If $2T < \Lambda$, then by Theorem 6, we know that the worst-case bound of the FFD-LS algorithm is not greater than 2 either, and we are also done. This complete the proof of Case 2.

Case 3 ($k \ge 4$). Clearly, in this case, we have $\Lambda > 2T$. Again, by Theorem 6, we know that the worst-case bound of the FFD-LS algorithm is not greater than 2, and we are done.

This completes the proof. □

3. A Modified Approximation Algorithm and Its Worst-Case Bound for $P_{(b>1)}$

We know from Example 1 and Theorem 8 that the worst-case bound of the FFD-LS algorithm can be arbitrarily large if $k = 2$ and is 2 otherwise. Following the notation in Xu et al. [14], $\mathcal{J}_i$ is the hybrid job which consists of the ith nonempty bin obtained by the FFD algorithm and the corresponding maintenance activity. Note that $\mathcal{J}_1$ is scheduled before $\mathcal{J}_2$ in the FFD-LS schedule. So we have $\lambda_1 = \text{level}(B_1)$ and $\lambda_2 = \text{level}(B_2)$. In order to avoid the arbitrarily large worst-case bound, as we observe, we only need to add the following step after the FFD-LS algorithm.

Step 4. If $k = 2$ and $\text{level}(B_2) < \text{level}(B_1)$, interchange $\mathcal{J}_1$ with $\mathcal{J}_2$.

Let us call this modified algorithm FFD-LS2I, which applies algorithm FFD-LS first and then applies Step 4. Clearly, the computational complexity of the FFD-LS2I is $O(n^2)$. Let the FFD-LS2I schedule be denoted by $S^{\star}$ and let the total processing times of the jobs in the ith batch $B_i^{\star}$ of $S^{\star}$ be denoted by $\lambda_i^{\star}$. It is easy to see that the number of the nonempty batches in $S^{\star}$ is also k because the fourth step of the FFD-LS2I algorithm does not change the number of batches.

In order to show that the worst-case bound of the FFD-LS2I algorithm is 2, by the structure of the FFD-LS2I algorithm and Theorem 8, we only need to consider the case $k = 2$.

Theorem 9. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS2I algorithm is not greater than 2 if $k = 2$.*

Proof. Clearly, $k = 2$ implies $T < \Lambda \le 2T$. Hence, by Lemma 4, we have $2 \le k^{*} \le 3$. Note that the FFD-LS2I schedule has two nonempty batches and that Lemma 3 indicates that the makespan of a feasible schedule with two batches is always less than that of a feasible schedule with three batches. So we have $k^{*} = 2$.

Substituting $k^* = 2$ into (1), we have

$$C^*_{\max} = T + a + b\left(\Lambda - \lambda^*_2\right) + \lambda^*_2. \tag{28}$$

Similar to (2), the makespan of the FFD-LS2I schedule is

$$C^{\text{FFD-LS2I}}_{\max} = T + a + b\left(\Lambda - \lambda^\star_2\right) + \lambda^\star_2. \tag{29}$$

Combining these two equations, we have

$$C^{\text{FFD-LS2I}}_{\max} = 2C^*_{\max} - T - a - b\left(\Lambda + \lambda^\star_2 - 2\lambda^*_2\right) + \left(\lambda^\star_2 - 2\lambda^*_2\right). \tag{30}$$

Note that $\lambda^*_1 + \lambda^*_2 = \Lambda > T$. We claim that $\lambda^*_2 > T/2$ for otherwise we have $\lambda^*_1 > T/2$ and thus we can get a schedule with a smaller makespan by interchanging all the jobs in the first batch of S^* with those in the second batch of S^*, which clearly violates the optimality of S^*. Combining this with $\lambda^\star_2 \le T$, we have

$$\lambda^\star_2 - 2\lambda^*_2 \le \lambda^\star_2 - T \le 0. \tag{31}$$

Case 1 ($4T/3 \le \Lambda \le 2T$). Note that $\lambda^\star_1 \le \lambda^\star_2$ and that $\lambda^\star_1 + \lambda^\star_2 = \Lambda$. So we have $\lambda^\star_2 \ge 2T/3$. And thus $\Lambda + \lambda^\star_2 - 2\lambda^*_2 \ge 2T - 2\lambda^*_2$. Combining this with $\lambda^*_2 \le T$, we have

$$\Lambda + \lambda^\star_2 - 2\lambda^*_2 \ge 0. \tag{32}$$

Substituting (31) and (32) into (30), we have

$$C^{\text{FFD-LS2I}}_{\max} \le 2C^*_{\max} - T - a, \tag{33}$$

which implies that $C^{\text{FFD-LS2I}}_{\max} < 2C^*_{\max}$.

Case 2 ($T < \Lambda < 4T/3$). Consider the schedule $S^\diamond$ which schedules the hybrid job $\mathcal{J}_2$ first and then the hybrid job $\mathcal{J}_1$. Let the total processing time of the jobs in the first batch of $S^\diamond$ and in the second batch of $S^\diamond$ be denoted by $\lambda^\diamond_1$ and $\lambda^\diamond_2$, respectively. Clearly, $\lambda^\diamond_1 + \lambda^\diamond_2 = \Lambda$. Note that Step 4 schedules the hybrid job with the highest level last. So we have $\lambda^\diamond_2 \le \lambda^\star_2$.

Let the makespan of the schedule $S^\diamond$ be denoted by $C^\diamond_{\max}$. It is easy to see that

$$C^\diamond_{\max} = T + a + b\left(\Lambda - \lambda^\diamond_2\right) + \lambda^\diamond_2. \tag{34}$$

Recall that $\lambda^\diamond_2 \le \lambda^\star_2$ and $b > 1$. Comparing (29) with (34), we have

$$C^{\text{FFD-LS2I}}_{\max} \le C^\diamond_{\max}. \tag{35}$$

Now, consider two subcases.

Subcase 2.1. There is only one job, say job J_u, in the first batch of the schedule $S^\diamond$. Without loss of generality, we assume that the jobs in $\mathcal{J}_1$ are ordered in nondecreasing order of their processing times; that is, $p^{(1)}_1 \ge p^{(1)}_2 \ge \cdots \ge p^{(1)}_{k_1}$. Suppose that $p^{(1)}_s \ge p_u > p^{(1)}_{s+1}$. Clearly, we have

$$\sum_{i=1}^{s} p^{(1)}_i + p_u > T, \tag{36}$$

$$\lambda^\diamond_2 = \Lambda - p_u. \tag{37}$$

Now, consider the position of job J_u in the optimal schedule S^*. If J_u is in the first batch of S^*, then we have $\lambda^*_2 \le \Lambda - p_u$. Combine this with (37), we have $\lambda^\diamond_2 \ge \lambda^*_2$. If J_u is in the second batch of S^*, then at least one job, say $J^{(1)}_v$ with $1 \le v \le s$, must be assigned to the first batch of S^*. To see this is the case, suppose all the jobs $J^{(1)}_1, \ldots, J^{(1)}_s$ are assigned to the second batch of S^*. We thus have $\lambda^*_2 \ge \sum_{i=1}^{s} p^{(1)}_i + p_u$. Combining this with (36), we have $\lambda^*_2 > T$, which implies that S^* is infeasible, clearly, a contradiction. Now, suppose $J^{(1)}_v$ is in the first batch of S^*, where $1 \le v \le s$. Note that $p^{(1)}_v \ge p_u$. So we have $\lambda^*_2 \le \Lambda - p^{(1)}_v \le \Lambda - p_u$. Combining this with (37), we have $\lambda^\diamond_2 \ge \lambda^*_2$. So, in either case, we have $\lambda^\diamond_2 \ge \lambda^*_2$. Combining this with (28) and (34), we have $C^\diamond_{\max} \le C^*_{\max}$. Combining this with (35), we have $C^{\text{FFD-LS2I}}_{\max} \le C^*_{\max}$, which implies that $C^{\text{FFD-LS2I}}_{\max} = C^*_{\max}$ by the optimality of S^*.

Subcase 2.2. There are at least two jobs in the first batch of the schedule $S^\diamond$. Let J_l be the job with the smallest processing time in the first batch of the schedule $S^\diamond$. Clearly, we have

$$\begin{aligned} \lambda^\diamond_1 &\ge 2p_l, \\ \lambda^\diamond_2 + p_l &> T \ge \lambda^*_2. \end{aligned} \tag{38}$$

Combining (28) with (34), we have

$$C^\diamond_{\max} = 2C^*_{\max} - T - a - b\left(\Lambda + \lambda^\diamond_2 - 2\lambda^*_2\right) + \left(\lambda^\diamond_2 - 2\lambda^*_2\right). \tag{39}$$

By (38), we know that

$$\Lambda + \lambda^\diamond_2 - 2\lambda^*_2 \ge \Lambda - \lambda^\diamond_2 - 2p_l. \tag{40}$$

Recall that $\Lambda - \lambda^\diamond_2 = \lambda^\diamond_1$ and that $\lambda^\diamond_1 \ge 2p_l$. So we have

$$\Lambda + \lambda^\diamond_2 - 2\lambda^*_2 \ge 0. \tag{41}$$

By a similar reasoning as in the arguments of deducing (31), we have

$$\lambda^\diamond_2 - 2\lambda^*_2 \le \lambda^\diamond_2 - T \le 0. \tag{42}$$

Substituting (41) and (42) into (39), we have

$$C^\diamond_{\max} \le 2C^*_{\max} - T - a, \tag{43}$$

which implies that $C^\diamond_{\max} \le 2C^*_{\max}$. Combining this with (35), we have $C^{\text{FFD-LS2I}}_{\max} \le 2C^*_{\max}$.

This completes the proof. □

Now, we are ready to present the worst-case bound of the FFD-LS2I algorithm.

Theorem 10. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS2I algorithm is 2 and the bound is tight.*

Proof. If $k \ne 2$, it is clear that the FFD-LS2I algorithm is reduced to the FFD-LS algorithm, and thus, by Theorem 8, we know that the worst-case bound of the FFD-LS2I algorithm

is not greater than 2. If $k = 2$, then, by Theorem 9, we know that the worst-case bound of the FFD-LS2I algorithm is not greater than 2 either. Hence, we have completed the proof that the worst-case bound of the FFD-LS2I algorithm is not greater than 2.

To show that this bound cannot be smaller than 2, consider the following instance. Let $T = 12$, $p_1 = 8 - 2/b$, $p_2 = p_3 = 4+1/b$, $p_4 = 4$, $p_5 = 4-2/b$, $p_6 = 2/b$, and $a = 1$. It is easy to see that $C_{\max}^{\text{FFD-LS2I}} = 24 + 24b + 2/b$ while $C_{\max}^{*} = 25 + 12b$. It follows that $C_{\max}^{\text{FFD-LS2I}}/C_{\max}^{*} = (24 + 24b)/(25 + 12b) \to 2$ as $b \to \infty$.

This completes the proof. □

4. Nonapproximability

In this section, we shall show that it is impossible to have a polynomial time approximation algorithm with a worst-case bound of less than 2 for the scheduling problem $P_{(b\geq 0)}$ unless P = NP, which implies that the proposed algorithm FFD-LS2I is the best possible approximation algorithm for $P_{(b>1)}$ from the worst-case bound point of view and that the FFD-LS algorithm proposed by Xu et al. [14] is the best possible approximation algorithm for $P_{(b\leq 1)}$ from the same point of view.

Note that the case $b = 0$ has been considered in Ji et al. [3] (see the following lemma, that is, Lemma 11), so we only need to consider the case $b > 0$, that is, the problem $P_{(b>0)}$.

Lemma 11 (Ji et al. [3]). *There is no polynomial time approximation algorithm with a worst-case bound less than 2 for the scheduling problem $P_{(b=0)}$ unless $P = NP$.*

The underlying idea of our approach for problem $P_{(b>0)}$ is by contradiction. Specifically, we shall show that if there exists an approximation algorithm with a worst-case bound of $2-\epsilon$ with $0 < \epsilon < 1$ for $P_{(b>0)}$ then it can be used to establish a polynomial time algorithm for the well-known NP-complete problem Partition. This clearly leads to a contradiction if $P \neq$ NP. Hence, such an approximation algorithm for the scheduling problem $P_{(b>0)}$ cannot exist unless P = NP.

Partition Problem (see, e.g., Garey and Johnson [19]). Given n positive integers $h_1, h_2, \ldots, h_n$ with $\sum_{i=1}^{n} h_i = 2H$, does there exist a set $X \subseteq \{1, 2, \ldots, n\}$ with $\sum_{i\in X} h_i = H$?

For any fixed positive number $\epsilon < 1$ and an instance I of the Partition problem, we construct an instance II of the scheduling problem $P_{(b>0)}$ as follows. There are n jobs $J_1, J_2, \ldots, J_n$ and the processing time of job J_i is $p_i = h_i$. Let $T = H$ and $a = (2T + bT)(1 - \epsilon)/\epsilon + 1$. It is clear that this construction can be performed in polynomial time.

Theorem 12. *If there exists a polynomial time approximation algorithm A_ϵ with a worst-case bound of $2-\epsilon$ for some positive number $\epsilon < 1$ for the scheduling problem $P_{(b>0)}$, then there exists a polynomial time algorithm for the Partition problem.*

Proof. Given any instance I of the Partition problem, we construct the corresponding instance II of the scheduling problem $P_{(b>0)}$ in polynomial time as stated above. Let $C_{\max}^{A_\epsilon}$ and $C_{\max}^{*}$ be the makespan of the schedule produced by the approximation algorithm A_ϵ and the makespan of an optimal schedule S^* for the instance II of $P_{(b>0)}$, respectively. We shall show that the instance I of the Partition problem can be answered by merely comparing the values of $C_{\max}^{A_\epsilon}$ and $2(T + a) + bT$.

To see that this is the case, let us apply algorithm A_ϵ to instance II. We claim that if $C_{\max}^{A_\epsilon} \leq 2(T + a) + bT$ then there exists a solution to instance I of the Partition problem and, otherwise, there does not exist a solution to instance I. In what follows, we shall show that this claim is correct.

Case 1 ($C_{\max}^{A_\epsilon} \leq 2(T+a)+bT$). Clearly, we have $C_{\max}^{*} \leq 2(T + a) + bT$ because $C_{\max}^{*} \leq C_{\max}^{A_\epsilon}$. Let k^* be the total number of batches used in S^*. Note that $T = H$ and $\Lambda = \sum_{i=1}^{n} p_i = 2H$, so we have $k^* \geq 2$.

Let λ_i^* be the total processing time of the jobs in the ith batch of S^*. If $k^* \geq 3$, then, by (1), we have $C_{\max}^{*} \geq 2(T + a) + b(\Lambda - \lambda_{k^*}^{*}) + \lambda_{k^*}^{*}$. Note that $\Lambda - \lambda_{k^*}^{*} = \sum_{i=1}^{k^*-1} \lambda_i^* \geq \lambda_1^* + \lambda_2^* > T$. So we have $C_{\max}^{*} > 2(T + a) + bT$, which contradicts the inequality $C_{\max}^{*} \leq 2(T+a)+bT$. This implies that $k^* = 2$. That is, the optimal schedule S^* has two nonempty batches. Now, let X be the set of indices of the jobs scheduled in the first batch of the optimal schedules. Clearly we have $\sum_{i\in X} p_i = T$. Recall that $T = H$ and $p_i = h_i$ for $i = 1, 2, \ldots, n$. So we have $\sum_{i\in X} h_i = H$, which implies that there exists a solution to the instance I of the Partition problem.

Case 2 ($C_{\max}^{A_\epsilon} > 2(T + a) + bT$). Recall that the worst-case bound of the algorithm A_ϵ is $2 - \epsilon$. So we have

$$C_{\max}^{*} \geq \frac{C_{A_\epsilon}}{2-\epsilon} > \frac{2(T + a) + bT}{2-\epsilon}. \tag{44}$$

Recall that $a = (2T + bT)(1 - \epsilon)/\epsilon + 1$. So we have

$$\epsilon = \frac{2T + bT}{2T + bT + a - 1}. \tag{45}$$

Substituting (45) into (44), we have

$$\begin{aligned} C_{\max}^{*} &> \frac{2(T + a) + bT}{2 - (2T + bT)/(2T + bT + a - 1)} \\ &= 2T + bT + a + \frac{2T + bT}{2T + bT + 2a - 2} \geq 2T + bT + a. \end{aligned} \tag{46}$$

If there exists a solution, namely, set X, to the instance I of the Partition problem, let the jobs whose indices belong to the set X be scheduled into the first batch and the remaining jobs the second batch; we then obtain a schedule with a makespan of $2T + a + bT$, which clearly contradicts (46). Hence, there does not exist a solution to the instance I of the Partition problem.

This completes the proof. □

By Theorem 10 and the fact that no NP-complete problem can be solved by a polynomial time algorithm unless P = NP, we have the following result.

Theorem 13. *There is no polynomial time approximation algorithm with a worst-case bound less than 2 for the scheduling problem $P_{(b>0)}$ unless $P = NP$.*

Combining Lemma 11 and Theorem 13, we have the following result.

Theorem 14. *There is no polynomial time approximation algorithm with a worst-case bound less than 2 for the scheduling problem $P_{(b\geq 0)}$ unless $P = NP$.*

Recall that we have showed in Theorem 10 that the worst-case bound of the FFD-LS2I algorithm is 2 for $P_{(b>1)}$. Combining this with Theorem 14, we have the following result.

Theorem 15. *The FFD-LS2I algorithm is the best possible polynomial time approximation algorithm for $P_{(b>1)}$ unless $P = NP$ from the worst-case bound point of view.*

Recall that Xu et al. [14] have showed that the worst-case bound of the FFD-LS algorithm is 2 for $P_{(b\leq 1)}$. Combining this with Theorem 14, we have the following result.

Theorem 16. *The FFD-LS algorithm is the best possible polynomial time approximation algorithm for $P_{(b\leq 1)}$ unless $P = NP$ from the worst-case bound point of view.*

5. An Answer to Another Open Problem

Viewing each batch as a bin, in Remark 3 of Section 4 of the paper of Xu et al. [14], Xu et al. claim that "it seems that it is not necessary for an optimal schedule of 1, $MS[T, T], T_M(t) = a + bt, b > 2 + a/T \parallel C_{\max}$ to have the minimum number of bins. Whether this is true may be an interesting problem for further study." The following example gives an answer to this open problem.

Example 17. Consider the following instance for the scheduling problem under consideration: $T = 20$, $a = 1$, $b = 12$, $n = 8$, $p_1 = p_2 = p_3 = p_4 = 13$, and $p_5 = p_6 = p_7 = p_8 = 5$. It is easy to see that the minimum number of bins for this instance is 4; see, for example, that job J_i and job J_{i+4} are packed into bin B_i for $i = 1, \ldots, 4$. The corresponding makespan is 729. However, if we assign job J_i to the ith bin for $i = 1, \ldots, 4$ and assign the remaining jobs to the fifth bin, we get an optimal schedule with a makespan of 728. Clearly, this implies that it is not necessary for an optimal schedule of the considered scheduling problem to have the minimum number of bins.

6. Conclusion

In this paper, we revisited the single machine scheduling problem considered in Xu et al. [14]; we provided an approximation named FFD-LS2I with a worst-case bound of 2 for the case $b > 1$. We showed that the FFD-LS2I algorithm is the best possible algorithm for the case $b > 1$ and that the FFD-LS algorithm is the best possible algorithm for the case $b \leq 1$ both from the worst-case bound point of view. We also give an answer to another open problem proposed in Xu et al. [14].

Future research may focus on the design of approximation algorithms with a lower time complexity while maintaining the worst-case bound of 2. The other maintenance duration functions such as convex function and concave function are also worth considering.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (71201022) and the Natural Science Foundation of Jiangxi Province (20122BAB201010).

References

[1] G. Schmidt, "Scheduling with limited machine availability," *European Journal of Operational Research*, vol. 121, no. 1, pp. 1–15, 2000.

[2] C.-Y. Lee, "Machine scheduling with availability constraints," in *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, J. Y.-T. Leung, Ed., pp. 22.1–22.13, CRC Press, Boca Raton, Fla, USA, 2004.

[3] M. Ji, Y. He, and T. C. E. Cheng, "Single-machine scheduling with periodic maintenance to minimize makespan," *Computers & Operations Research*, vol. 34, no. 6, pp. 1764–1770, 2007.

[4] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 199–211, 2010.

[5] K. Sun and H. Li, "Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines," *International Journal of Production Economics*, vol. 124, no. 1, pp. 151–158, 2010.

[6] C.-J. Hsu, T. C. E. Cheng, and D.-L. Yang, "Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time," *Information Sciences*, vol. 181, no. 20, pp. 4799–4803, 2011.

[7] S. Bock, D. Briskorn, and A. Horbach, "Scheduling flexible maintenance activities subject to job-dependent machine deterioration," *Journal of Scheduling*, vol. 15, pp. 565–578, 2012.

[8] V. Gordon, V. Strusevich, and A. Dolgui, "Scheduling with due date assignment under special conditions on job processing," *Journal of Scheduling*, vol. 15, pp. 447–456, 2012.

[9] B. Mor and G. Mosheiov, "Scheduling a maintenance activity and due-window assignment based on common flow allowance," *International Journal of Production Economics*, vol. 135, no. 1, pp. 222–230, 2012.

[10] D. Xu, M. Liu, Y. Yin, and J. Hao, "Scheduling tool changes and special jobs on a single machine to minimize makespan," *Omega*, vol. 41, pp. 299–304, 2013.

[11] D. Xu and D.-L. Yang, "Makespan minimization for two parallel machines scheduling with a periodic availability constraint: mathematical programming model, average-case analysis, and anomalies," *Applied Mathematical Modelling*, vol. 37, pp. 7561–7567, 2013.

[12] D.-L. Yang and S.-J. Yang, "Unrelated parallel-machine scheduling with multiple rate-modifying activities," *Information Sciences*, vol. 235, pp. 280–286, 2013.

[13] S.-J. Yang, "Unrelated parallel-machine scheduling with deterioration effects and deteriorating multimaintenance activities for minimizing the total completion time," *Applied Mathematical Modelling*, vol. 37, pp. 2995–3005, 2013.

[14] D. Xu, Y. Yin, and H. Li, "Scheduling jobs under increasing linear machine maintenance time," *Journal of Scheduling*, vol. 13, no. 4, pp. 443–449, 2010.

[15] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[16] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, Ed., pp. 46–93, PWS, Boston, Mass, USA, 1997.

[17] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, New York, NY, USA, 4th edition, 2012.

[18] D. Simchi-Levi, "New worst-case results for the bin-packing problem," *Naval Research Logistics*, vol. 41, no. 4, pp. 579–585, 1994.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, NY, USA, 1979.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 576234, 6 pages
http://dx.doi.org/10.1155/2014/576234

Research Article

Semi-Online Scheduling on Two Machines with *GoS* Levels and Partial Information of Processing Time

Taibo Luo[1] and Yinfeng Xu[1,2]

[1] *Business School, Sichuan University, Chengdu 610065, China*
[2] *State Key Lab for Manufacturing Systems Engineering, Xi'an 710049, China*

Correspondence should be addressed to Yinfeng Xu; yfxu@scu.edu.cn

Received 15 November 2013; Accepted 19 December 2013; Published 11 February 2014

Academic Editors: J. G. Barbosa and D. Oron

Copyright © 2014 T. Luo and Y. Xu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper investigates semi-online scheduling problems on two parallel machines under a grade of service (*GoS*) provision subject to minimize the makespan. We consider three different semi-online versions with knowing the total processing time of the jobs with higher *GoS* level, knowing the total processing time of the jobs with lower *GoS* level, or knowing both in advance. Respectively, for the three semi-online versions, we develop algorithms with competitive ratios of 3/2, 20/13, and 4/3 which are shown to be optimal.

1. Introduction

Scheduling problem under a grade of service provision was first proposed by Hwang et al. [1]. The jobs should be assigned irrevocably to machines as soon as they arrive. Each job and machine are labeled with the *GoS* levels. A job can be processed by a particular machine if and only if the *GoS* level of the job is not less than that of the machine. In practice, the scheduling with *GoS* eligibility constraints is widely used. For more details, please refer to [1–3].

For the offline scheduling on m machines under a grade of service provision, Hwang et al. [1] presented an LG-LPT algorithm which has a tight bound of 5/4 for two machines and $2 - 1/(m-1)$ for m ($m \geq 3$) machines. Ji and Cheng [4] gave an FPTAS for this problem. However, Woeginger [5] gave two simpler FPTASs for the same problem. For the online version, Jiang [6] proved a lower bound of 2 for the case with two levels of *GoS* and presented an online algorithm with a competitive ratio of 2.52. Zhang et al. [7] improved the result and showed an algorithm with a competitive ratio of $1 + (m^2 - m)/(m^2 - sm + s^2) \leq 7/3$.

For $m = 2$, Park et al. [8] presented an optimal algorithm with a competitive ratio of 5/3. However, there are many papers focusing on the semi-online scheduling on two machines under *GoS* [8–13]. As the total processing time of all jobs is known, Park et al. [8] gave an optimal algorithm with a competitive ratio of 3/2. When the largest processing time of all jobs is known, Wu et al. [12] presented an optimal algorithm with a competitive ratio of $(\sqrt{5}+1)/2$. In the same paper, Wu et al. [12] gave an optimal algorithm with a competitive ratio of 3/2 when the optimal value of the instance is known. For the processing times bounded in an interval, Liu et al. [9] gave a competitive algorithm under some conditions and Zhang et al. [13] improved the result and gave an optimal algorithm for different intervals. In this paper, we focus on the semi-online scheduling problem where only partial information of the total processing time is known. The semi-online versions concerned in our paper are listed as follows.

sum · higher: the total processing time of all jobs with higher *GoS* level is known in advance.

sum · lower: the total processing time of all jobs with lower *GoS* level is known in advance.

We use $P2 \cdot GoS \mid s \mid C_{\max}$ to denote the semi-online problem with information s, where $s \in \{sum \cdot higher, sum \cdot lower\}$. Moreover, we use $P2 \cdot GoS \mid s1\&s2 \mid C_{\max}$ to denote the semi-online problem where both information $s1$ and information $s2$ are available in advance.

Our results indicate that competitive ratios of three algorithms are better than 5/3 of the online version [8]. When knowing *sum · higher* in advance, the competitive ratio is

the same as the optimal algorithm [8] presented for the semi-online problem where the total processing time of all jobs is known. For designing algorithm, the results indicate that knowing *sum · higher*&*sum · lower* is more useful than *sum · higher* or *sum · lower*. Moreover, knowing *sum · higher* is more useful than *sum · lower*. However, knowing *sum · lower* is better than knowing the largest processing time of all jobs [12].

The rest of this paper is organized as follows: in Section 2, we give some basic definitions. In Sections 3–5, we prove lower bounds and present algorithms for the three semi-online problems, respectively.

2. Basic Definitions

We are given two machines and a series of jobs arriving online which are to be scheduled irrevocably at the time of their arrivals. The arrival of a new job occurs only after the current job is scheduled. We denote by the set $J = 1, \ldots, n$ the set of all job indices arranged in the order of arrival. The jth arriving job is referred to as job J_j. We denote each job by $J_j = (p_j, g_j)$. The *GoS* assigned to job J_j is denoted by g_j, which is 1 if the job must be processed only by the first machine and 2 if it may be processed by either of the two; p_j is the processing time of job J_j; p_j and g_j are not known until the arrival of job J_j. Let $T_1 = \sum_{g_j=1} p_j$ and $T_2 = \sum_{g_j=2} p_j$.

The scheduled can be seen as the partition of J into two subsets, denoted by (S_1, S_2), where S_1 and S_2 contain job indices assigned to the first and second machine, respectively. Let $t(S) = \Sigma_{J_j \in S} p_j$ for an arbitrary subset S of J. Then the maximum of $t(S_1)$ and $t(S_2)$, denoted by $\max\{t(S_1), t(S_2)\}$, is the makespan of the scheduled (S_1, S_2). The problem is to minimize $\max\{t(S_1), t(S_2)\}$.

The minimum makespan obtained by an optimal offline algorithm is denoted by C_{opt}. The makespan generated by algorithm A is denoted by C_A. The competitive ratio of A is defined to be the maximum C_A/C_{opt}.

3. Optimal Algorithm for $P2 \cdot GoS \mid sum \cdot higher \mid C_{\max}$

In this section, we prove a lower bound of competitive ratio and present an optimal algorithm for the semi-online version as T_1 is known in advance.

3.1. Lower Bound of Competitive Ratio

Theorem 1. *Any semi-online algorithm for* $P2 \cdot GoS \mid sum \cdot higher \mid C_{\max}$ *has a competitive ratio of at least* 3/2.

Proof. The theorem will be proved by adversary method. Let $T_1 = 1$ be known in advance. The first job is $J_1 = (1, 1)$. J_1 must be scheduled on the first machine. Then, we generate job $J_2 = (1, 2)$. If job $J_2 = (1, 2)$ is scheduled on the first machine, and there is no job arriving after, we have $C_A/C_{\text{opt}} = 2 > 3/2$. Otherwise, job J_2 is scheduled on the second machine, and we generate job $J_3 = (2, 2)$. No matter which machine that job J_3 is scheduled on, we have $C_A = 3$. Since the optimal algorithm will scheduled job J_1 and job J_2 on the first machine and scheduled job J_3 on the second machine. Hence, $C_A/C_{\text{opt}} = 3/2$. The proof is completed. □

3.2. Optimal Semi-Online Algorithm GoS-TH.

Since we know T_1 in advance and all the jobs with $g_j = 1$ must be scheduled on the first machine, we can regard them as one job; that is, $J_0 = (T_1, 1)$. We scheduled job J_0 on the first machine at first and do not need to care about the job with $g_j = 1$ later.

At the arrival of each job, H is updated to become a half of the total processing time of the jobs which include the jobs with $GoS = 2$ and job J_0; P is updated to become the maximum processing time. We define P^j, S_1^j, S_2^j, and H^j to be P, S_1, S_2, and H after we scheduled job J_j. Then, clearly the optimum makespan $C_{\text{opt}} \geq L = \max(T, P)$. Combined with the online algorithm presented by Park et al. [8], we propose Algorithm *GoS-TH*.

Algorithm GoS-TH

(1) Scheduled J_0 to the first machine and $t(S_1^0) = T_1$.

(2) Let $p_j = 0$ for all the jobs with $g_j = 1$.

(3) Suppose that the incoming job is J_j and $g_j = 1$; assign job J_j to the first machine.

(4) Suppose the incoming job is J_j and $g_j = 2$. $H_j = H_{j-1} + p_j/2$, $P_j = \max(P_{j-1}, p_j)$, and $L_j = \max(T_j, P_j)$.

(4.1) If $t(S_2^j) + p_j \leq (3/2)L_j$, scheduled it to the second machine.

(4.2) If $t(S_2^j) + p_j > (3/2)L_j$, scheduled it to the first machine.

Theorem 2. *The competitive ratio of Algorithm GoS-TH is* 3/2 *for* $P2 \cdot GoS \mid sum \cdot higher \mid C_{\max}$.

Proof. Suppose that Theorem 2 is false. There must exist an instance with the least number of jobs to make $C_{GoS\text{-}TH} > (3/2)C_{\text{opt}}$. The makespan is not determined until the arrival of job J_n. Therefore,

$$C_{GoS\text{-}TH} = \max\left\{t\left(S_1^n\right), t\left(S_2^n\right)\right\} > \frac{3}{2}C_{\text{opt}}, \quad \max\left\{t\left(S_1^{n-1}\right), t\left(S_2^{n-1}\right)\right\} \leq \frac{3}{2}C_{\text{opt}}. \tag{1}$$

If $g_n = 1$, based on Algorithm *GoS-TH*, we have $t(S_1^n) = t(S_1^{n-1})$ and $t(S_2^n) = t(S_2^{n-1})$. Due to the fact that $\max\{t(S_1^{n-1}), t(S_2^{n-1})\} \leq (3/2)C_{\text{opt}}$, we just need to prove Theorem 2 is true when $g_n = 2$.

If $g_n = 2$, when it is scheduled on the second machine, we have $C_{GoS\text{-}TH} = t(S_2^n) = t(S_2^{n-1}) + p_n \leq (3/2)C_{\text{opt}}$. This is contradicting with inequality (1). Otherwise, job J_n is scheduled on the first machine, which leads to $t(S_2^{n-1}) + p_n > (3/2)L_n \geq (3/2)H_n$; combined with $(t(S_2^{n-1}) + p_n + t(S_1^{n-1}))/2 = H_n$, we get that $t(S_1^{n-1}) < (1/2)L_n \leq (1/2)C_{\text{opt}}$. Since $p_n \leq C_{\text{opt}}$, we have $t(S_1^{n-1}) + p_n < (3/2)C_{\text{opt}}$, which also contradicted with inequality (1). The proof is completed. □

4. Optimal Algorithm for $P2 \cdot GoS \mid sum \cdot lower \mid C_{\max}$

In this section, we prove a lower bound of competitive ratio and present an optimal algorithm for the semi-online version as T_2 is known in advance.

4.1. Lower Bound of Competitive Ratio

Theorem 3. *Any semi-online algorithm for $P2 \cdot GoS \mid sum \cdot lower \mid C_{\max}$ has a competitive ratio of at least* 20/13.

Proof. We will construct a job sequence with $T_2 = 26$ to make an arbitrary algorithm A behave poorly. We begin with jobs $J_1 = (1, 2)$ and $J_2 = (1, 2)$. We discuss the following three cases.

Case 1 (J_1 and J_2 are scheduled on the first machine). We continue to generate jobs $J_3 = (12, 2)$ and $J_4 = (12, 2)$. If job J_3 and job J_4 are scheduled on the second machine, we have $t(S_2) = 24$; thus, $C_A/C_{\text{opt}} = 24/14 > 20/13$. Otherwise, if job J_3 or job J_4 is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job $J_5 = (26, 1)$. We will have $t(S_1) \geq 40$. Since optimal algorithm will scheduled jobs J_1, J_2, J_3, and J_4 on the second machine and scheduled job J_5 on the first machine, we have $C_A/C_{\text{opt}} \geq 20/13$.

Case 2 (J_1 or J_2 is scheduled on the first machine). We continue to generate job $J_3 = (2, 2)$. Then we discuss the following two subcases.

Subcase 2.1 (job J_3 is scheduled on the first machine). We continue to generate jobs $J_4 = (11, 2)$ and $J_5 = (11, 2)$. If job J_4 and job J_5 are scheduled on the second machine, we have $t(S_2) = 23$ and $t(S_1) = 2$, which lead to $C_A/C_{\text{opt}} = 23/13 > 20/13$. Otherwise, if job J_4 or job J_5 is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job $J_6 = (26, 1)$ which leads to $C_A/C_{\text{opt}} \geq 20/13$.

Subcase 2.2 (J_3 is scheduled on the second machine). We generate job $J_4 = (5, 2)$. If job J_4 is scheduled on the second machine, we generate job $J_5 = (13, 2)$. If job J_5 is scheduled on the second machine, we have $t(S_2) = 21$. Since the optimal algorithm will scheduled jobs J_1, J_2, J_3, and J_4 on the first machine and scheduled job J_5 on the second machine, we have $C_A/C_{\text{opt}} = 21/13 > 20/13$. If job J_5 is scheduled on the first machine, we generate job $J_6 = (26, 1)$ which leads to $C_A/C_{\text{opt}} \geq 20/13$.

Otherwise, if job J_4 is scheduled on the first machine, we generate job $J_5 = (8, 2)$. If job J_5 is scheduled on the first machine, we generate job $J_6 = (26, 1)$ which leads to $C_A/C_{\text{opt}} \geq 20/13$. If job J_5 is scheduled on the second machine, we generate job $J_6 = (9, 2)$. If job J_6 is scheduled on the second machine, we have $t(S_2) = 20$ and $C_{\text{opt}} = 13$; thus, $C_A/C_{\text{opt}} = 20/13$. If J_6 is scheduled on the first machine, we generate job $J_7 = (26, 1)$ and have $t(S_1) = 40$ and $C_{\text{opt}} = 26$, which also lead to $C_A/C_{\text{opt}} = 20/13$.

Case 3 (J_1 and J_2 are scheduled on the second machine). We continue to generate job $J_3 = (2, 2)$. Then we discuss the following two subcases.

Subcase 3.1 (job J_3 is scheduled on the second machine). We generate jobs $J_4 = (3, 2)$ and $J_5 = (3, 2)$. If job J_4 or job J_5 is scheduled on the second machine or both of them are scheduled on the second machine, we further generate job $J_6 = (1, 2)$. If job J_6 is scheduled on the second machine, we generate job $J_7 = (14, 2)$. If job J_7 is scheduled on the second machine, we have $t(S_2) = 22$. Since the optimal algorithm will scheduled jobs J_1, J_2, J_3, J_4, J_5, and J_6 on the first machine and scheduled job J_7 on the second machine, we have $C_A/C_{\text{opt}} \geq 22/14 > 20/13$. Otherwise, job J_7 is scheduled on the first machine, and then we generate job $J_8 = (26, 1)$ and have $t(S_1) \geq 40$ and $C_{\text{opt}} = 26$; hence, $C_A/C_{\text{opt}} > 20/13$. If job J_6 is scheduled on the first machine, we generate job $J_7 = (13, 2)$. If job J_7 is scheduled on the second machine, we have $t(S_2) \geq 20$ and $C_{\text{opt}} = 13$, hence, we also have $C_A/C_{\text{opt}} > 20/13$. Otherwise, job J_7 is scheduled on the first machine, and then we generate job $J_8 = (26, 1)$. We have $C_A/C_{\text{opt}} > 20/13$.

If job J_4 and job J_5 are scheduled on the first machine, we generate jobs $J_6 = (8, 2)$ and $J_7 = (8, 2)$. If job J_6 or job J_7 is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job $J_8 = (26, 1)$. We have $C_A/C_{\text{opt}} > 20/13$. Otherwise, if job J_6 and job J_7 are scheduled on the second machine, we have $t(S_2) = 20$ and $C_{\text{opt}} = 13$, which lead to $C_A/C_{\text{opt}} = 20/13$.

Subcase 3.2 (job J_3 is scheduled on the first machine). We generate job $J_4 = (2, 2)$. If job J_4 is scheduled on the first machine, we generate jobs $J_5 = (10, 2)$ and $J_6 = (10, 2)$. If job J_5 and job J_6 are scheduled on the second machine, we have $t(S_2) = 22$ and $C_{\text{opt}} = 13$; thus, $C_A/C_{\text{opt}} = 22/13 > 20/13$. Otherwise, if job J_5 or job J_6 is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job $J_7 = (26, 1)$ and have $t(S_1) \geq 40$; thus, $C_A/C_{\text{opt}} = 20/13$.

Otherwise, if job J_4 is scheduled on the second machine, we generate job $J_5 = (4, 2)$. If job J_5 is scheduled on the second machine, we generate job $J_6 = (13, 2)$. If job J_6 is scheduled on the second machine, we have $t(S_2) = 21$ and $C_{\text{opt}} = 13$; hence, $C_A/C_{\text{opt}} = 20/13$. If job J_6 is scheduled on the first machine, we generate job $J_7 = (26, 1)$ and have $t(S_1) \geq 41$; thus, $C_A/C_{\text{opt}} = 41/26 > 20/13$.

Therefore, job J_5 is scheduled on the first machine. Then we generate jobs $J_6 = (8, 2)$ and $J_7 = (8, 2)$. If job J_6 or job J_7 is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job $J_8 = (26, 1)$. We will have $t(S_1) \geq 40$ and $C_{\text{opt}} = 26$; hence, $C_A/C_{\text{opt}} > 20/13$. Otherwise, if job J_6 and job J_7 are scheduled on the second machine, we have $t(S_2) = 20$ and $C_{\text{opt}} = 13$; also, we have $C_A/C_{\text{opt}} = 20/13$. The proof is completed. □

4.2. Optimal Semi-Online Algorithm GoS-TL. In this subsection, we design an optimal algorithm with a competitive ratio of 20/13. Let x_1 and x_2 be the jobs with $g_j = 2$ assigned

to the first and second machine, respectively, where $t(x_1) + t(x_2) = T_2$. We define x_1^j and x_2^j to be x_1 and x_2 after we scheduled job J_j. Then, we propose Algorithm *GoS-TL*.

Algorithm GoS-TL

(1) Suppose that the incoming job is J_j and $g_j = 1$; assign job J_j to the first machine.

(2) Suppose that the incoming job is J_j and $g_j = 2$.

(2.1) (*Stopping criterion 1*). If $(3/13)T_2 \le t(x_1^{j-1}) + p_j \le (7/13)T_2$, scheduled job J_j and all the remaining jobs with $GoS = 1$ to the first machine, and then scheduled all the remaining jobs with $GoS = 2$ to the second machine. Stop.

(2.2) (*Stopping criterion 2*). If $(6/13)T_2 \le t(x_2^{j-1}) + p_j \le (10/13)T_2$, scheduled job J_j to the second machine and scheduled all the remaining jobs to the first machine. Stop.

(2.3) (*Stopping criterion 3*). If $t(x_2^{j-1}) > 10/13$, scheduled job J_j and the remaining jobs to the first machine. Stop.

(2.4) If $t(x_2^{j-1}) \le (7/26)T_2$ and $(7/13)T_2 \le t(x_2^{j-1}) + p_j < (6/13)T_2$, scheduled job J_j to the first machine. Continue.

(2.5) Otherwise, scheduled job J_j to the second machine. Continue.

Lemma 4. *If* $(6/13)T_2 \le t(x_2) \le (10/13)T_2$, *then* $C_{GoS\text{-}TL} \le (20/13)C_{opt}$.

Proof. Since $C_{\text{opt}} \ge (1/2)T_2$, we have $t(S_2) = t(x_2) \le (10/13)T_2 \le (20/13)C_{\text{opt}}$. Then, we only need to prove that $t(S_1) \le (20/13)C_{\text{opt}}$. We discuss it by the following two cases.

Case 1 ($T_1 \ge T_2$). In this case, we have $t(S_1) = t(x_1) + T_1$ and $T_1 \ge (T_1 + T_2)/2$. Since $(6/13)T_2 \le t(x_2) \le (10/13)T_2$, we have $t(x_1) \le (7/13)T_2$. Combined with $C_{\text{opt}} \ge T_1$, we have

$$\frac{C_{GoS\text{-}TL}}{C_{\text{opt}}} \le \frac{(7/13)\,T_2 + T_1}{T_1} \le \frac{(7/13)\,T_1 + T_1}{T_1} = \frac{20}{13}. \quad (2)$$

Case 2 ($T_1 < T_2$). In this case, we have $T_1 < (T_1 + T_2)/2$, which leads to $C_{\text{opt}} \ge (T_1 + T_2)/2$. Then, we have

$$\frac{C_{GoS\text{-}TL}}{C_{\text{opt}}} \le \frac{(14/13)\,T_2 + 2T_1}{T_1 + T_2}. \quad (3)$$

Since $((14/13)T_2 + 2T_1)/(T_1 + T_2)$ is increasing function of the variation of T_1, thus, we have

$$\frac{C_{GoS\text{-}TL}}{C_{\text{opt}}} \le \frac{(14/13)\,T_2 + 2T_1}{T_1 + T_2} < \frac{(14/13)\,T_2 + 2T_2}{T_2 + T_2} = \frac{20}{13}. \quad (4)$$

The proof is completed. □

Based on Lemma 4, we straightforwardly have Corollary 5.

Corollary 5. (1) *If* $t(x_1) \le (7/13)T_2$, *then* $t(S_1) \le (20/13)C_{opt}$; (2) *if* $t(x_2) \le (10/13)T_2$, *then* $t(S_2) \le (20/13)C_{opt}$.

Lemma 6. $t(x_1) \le (7/13)T_2$.

Proof. Since Algorithm *GoS-TL* will only scheduled a job with $g_j = 2$ to the first machine only when $t(x_1^{j-1}) + p_j \le (7/13)T_2$, the lemma can directly be got from the Algorithm *GoS-TL*. The proof is completed. □

By using Corollary 5 and Lemma 6, we can obtain the following corollary.

Corollary 7. *If* $t(x_2) \le (10/13)T_2$, *then* $C_{GoS\text{-}TL} \le (20/13)C_{opt}$.

Based on Lemma 6 and Corollary 7, if we prove $t(S_2) \le (20/13)C_{\text{opt}}$ will hold when $t(x_2) > (10/13)T_2$, then we can prove that Algorithm *GoS-TL* is (20/13)-competitive.

Lemma 8. *If job* J_j *is scheduled on the second machine by Algorithm GoS-TL where* $t(x_2^{j-1}) + p_j > (10/13)T_2$ *and* $t(x_2^{j-1}) \le (7/26)T_2$, *then* $t(S_2) < (20/13)C_{opt}$.

Proof. If $t(x_2^{j-1}) + p_j > (10/13)T_2$ and $t(x_2^{j-1}) \le (7/13)T_2$, we have $p_j > (1/2)T_2$. Then we have $C_{\text{opt}} \ge p_j > (1/2)T_2$. If job J_j is scheduled on the second machine, Algorithm *GoS-TL* will scheduled the remaining jobs on the first machine. Thus,

$$\frac{t\left(S_2^j\right)}{C_{\text{opt}}} \le \frac{t\left(S_2^j\right)}{p_j} = \frac{t\left(x_2^{j-1}\right) + p_j}{p_j}. \quad (5)$$

Since $(t(x_2^{j-1}) + p_j)/p_j$ is decreasing function of the variation of p_j and increasing function of variation of $t(x_2^{j-1})$, we have

$$\frac{t\left(S_2^j\right)}{C_{\text{opt}}} \le \frac{t\left(x_2^{j-1}\right) + p_j}{p_j} < \frac{(7/26)\,T_2 + (1/2)\,T_2}{(1/2)\,T_2} = \frac{20}{13}. \quad (6)$$

The proof is completed. □

Lemma 9. *If* $(6/13)T_2 \le t(x_2^{j-1}) \le (10/13)T_2$, *then* $t(S_2) > (10/13)T_2$ *will never happen.*

Proof. If $(6/13)T_2 \le t(x_2^{j-1}) \le (10/13)T_2$, Algorithm *GoS-TL* will scheduled the remaining jobs on the first machine, so $t(S_2) = t(x_2^{j-1}) \le (10/13)T_2$. The proof is completed. □

Lemma 10. *If job* J_j *is scheduled on the second machine by Algorithm GoS-TL where* $(7/26)T_2 < t(x_2^{j-1}) < (6/13)T_2$ *and* $t(x_2^j) > (10/13)T_2$, *then* $t(S_2) = t(x_2^j) < (20/13)C_{opt}$.

Proof. Let J_w be the job that is scheduled on the second machine by Algorithm *GoS-TL* which satisfies $t(x_2^{w-1}) \le 7/13$ and $(7/26)T_2 < t(x_2^{w-1}) + p_w < (6/13)T_2$. Since J_w is

scheduled on the second machine by Algorithm *GoS-TL* and $(7/26)T_2 < t(x_2^{w-1}) + p_w < (6/13)T_2$, we have $t(x_1^{w-1}) + p_w > (7/13)T_2$; otherwise, Algorithm *GoS-TL* will scheduled job J_w on the first machine. Moreover, $t(x_1^{w-1}) < (3/13)T_2$ must hold which leads to $p_w > (4/13)T_2$. This further implies that $t(x_2^{w-1}) < (2/13)T_2$ since $t(x_2^{w-1}) + p_w < (6/13)T_2$.

If $t(x_1^{w-1}) = 0$, then $p_w > (7/13)T_2$ which leads to $t(x_2^w) > (6/13)T_2$; this is contradicting with the definition of job J_w. Therefore, we have $t(x_1^{w-1}) > 0$. Based on step (2.4) of Algorithm *GoS-TL*, the processing time of the job that is assigned to x_1^{w-1} is larger than $(3/26)T_2$ since $t(x_2^{w-1}) < (2/13)T_2$. Combined with $t(x_1^{w-1}) < (3/13)T_2$, we know there is only one job in x_1^{w-1}, call it job J_v. Since job J_v, job J_w, and $t(x_2^{w-1})$ need to satisfy

$$\begin{aligned} p_v + p_w &> \frac{7}{13}T_2, \\ p_v + t\left(x_2^{w-1}\right) &> \frac{7}{26}T_2, \\ \frac{7}{26}T_2 < t\left(x_2^{w-1}\right) + p_w &< \frac{6}{13}T_2, \end{aligned} \tag{7}$$

we can get $p_v > (9/52)T_2$, which implies that $t(S_2) < (43/52)T_2$.

Since there is no job with $g_j = 2$ that will scheduled on the first machine between job J_v and J_j by Algorithm *GoS-TL*, combined with job J_j being scheduled on the second machine, we have $p_v + p_j > (7/13)T_2$. Since $p_j > (4/13)T_2$, p_v, p_w, and p_j must satisfy

$$\begin{aligned} p_v + p_w &> \frac{7}{13}T_2, \\ p_v + p_j &> \frac{7}{13}T_2, \\ p_w + p_j &\geq \frac{9}{13}T_2, \end{aligned} \tag{8}$$

which implies that $C_{\text{opt}} > (7/13)T_2$, hence,

$$\frac{t\left(S_2\right)}{C_{\text{opt}}} < \frac{(43/52)\,T_2}{(7/13)\,T_2} = \frac{20}{13}. \tag{9}$$

The proof is completed. □

Based on Lemma 4 to Lemma 10 and Corollary 5 to Corollary 7, we have the following theorem naturally.

Theorem 11. *The competitive ratio of Algorithm GoS-TL is* 20/13 *for* $P2 \cdot GoS \mid sum \cdot lower \mid C_{\max}$.

5. Optimal Algorithm for $P2 \cdot GoS \mid sum \cdot high\&sum \cdot lower \mid C_{max}$

In this section, we show a lower bound of competitive ratio and present an optimal algorithm for the semi-online version as T_1 and T_2 are known in advance.

5.1. Lower Bound of Competitive Ratio

Theorem 12. *Any semi-online algorithm for* $P2 \cdot GoS \mid sum \cdot high\&sum \cdot lower \mid C_{\max}$ *has a competitive ratio of at least* 4/3.

Proof. The theorem will be proved by adversary method. Let $T_1 = 1/3$ and $T_2 = 5/3$ be known in advance. The first job is $J_1 = (1/3, 1)$. Job J_1 must be scheduled on the first machine. Then job $J_2 = (1/3, 2)$ arrives. If job J_2 is scheduled on the second machine, we further generate jobs $J_3 = (1, 2)$ and $J_4 = (1/3, 2)$. In this situation, we have $C_A \geq 4/3$ and $C_{\text{opt}} = 1$ since the optimal algorithm will scheduled jobs J_1, J_2, and J_4 on the first machine and scheduled job J_3 on the second machine. Thus, we have $C_A/C_{\text{opt}} \geq 4/3$. Otherwise, if job J_2 is scheduled on the first machine, then we further generate jobs $J_3 = (2/3, 2)$ and $J_4 = (2/3, 2)$. In this situation, we also have $C_A \geq 4/3$ and $C_{\text{opt}} = 1$ since the optimal algorithm will scheduled jobs J_1 and J_3 on the first machine and scheduled jobs J_2 and J_3 on the second machine. The proof is completed. □

5.2. Optimal Semi-Online Algorithm GoS-TB. In this subsection, we design an optimal algorithm with a competitive ratio of 4/3. Since we know T_1 in advance and all the jobs with $g_j = 1$ must be scheduled on the first machine, therefore, we can regard them as one job, that is, $J_0 = (T_1, 1)$. We scheduled job J_0 on the first machine at first and do not need to care about the job with $g_j = 1$ later. We present Algorithm *GoS-TB* as follows.

Algorithm GoS-TB

(1) Scheduled J_0 to the first machine and $t(S_1^0) = T_1$.
(2) Let $p_j = 0$ for all the jobs with $g_j = 1$.
(3) Suppose that the incoming job is J_j and $g_j = 1$; assign job J_j to the first machine.
(4) Suppose that the incoming job is J_j and $g_j = 2$.
  (4.1) If $t(S_1) + p_j \leq (2/3)(T_1 + T_2)$, scheduled it to the first machine.
  (4.2) (*Stopping criterion*). Suppose $t(S_1) + p_j > (2/3)(T_1 + T_2)$.
    (4.2.1) If $t(S_1^{j-1}) > (1/3)(T_1 + T_2)$, scheduled job J_j and the remaining jobs with $GoS = 2$ on the second machine and scheduled the remaining jobs with $GoS = 1$ to the first machine. Stop.
    (4.2.2) If $t(S_1^{j-1}) \leq (1/3)(T_1 + T_2)$, scheduled job J_j to the second machine, and scheduled the remaining jobs to the first machine. Stop.

Theorem 13. *The competitive ratio of Algorithm GoS-TH is* 4/3 *for* $P2 \cdot GoS \mid sum \cdot high\&sum \cdot lower \mid C_{\max}$.

Proof. Since the job with $g_j = 1$ is scheduled at first and we do not need to care about them after that. We focus on the jobs with $g_j = 2$.

Assume job J_j is the first job with $g_j = 2$ to make $t(S_1) + p_j > (2/3)(T_1 + T_2)$. We prove it by the following two cases.

Case 1 ($t(S_1^{j-1}) > (1/3)(T_1 + T_2)$). In this case, Algorithm *GoS-TB* schedules job J_j and the remaining jobs with $GoS = 2$ on the second machine. Therefore, $t(S_1) \le (2/3)(T_1 + T_2)$ and $t(S_2) \le T_1 + T_2 - t(S_1) < (2/3)(T_1 + T_2)$. Since $C_{\text{opt}} \ge (1/2)(T_1 + T_2)$, we have $C_{GoS\text{-}TH} = \max\{t(S_1), t(S_2)\} \le (2/3)(T_1 + T_2) \le (4/3)C_{\text{opt}}$.

Case 2 ($t(S_1^{j-1}) \le (1/3)(T_1 + T_2)$). In this case, we have $p_j > (1/3)(T_1 + T_2)$. Algorithm *GoS-TH* will only scheduled job J_j on the second machine and scheduled the remaining jobs on the first machine. If $p_j \ge (1/2)(T_1 + T_2)$, we have $C_{\text{opt}} = p_j$ and $t(S_1) \le p_j = t(S_2)$. Therefore, $C_{GoS\text{-}TH} = C_{\text{opt}}$. Otherwise, $(1/3)(T_1 + T_2) < p_j < (1/2)(T_1 + T_2)$, which leads to $t(S_1) \le T_1 + T_2 - t(S_2) < (2/3)(T_1 + T_2)$. Hence, we have $C_{GoS\text{-}TH} = \max\{t(S_1), t(S_2)\} \le (2/3)(T_1 + T_2) \le (4/3)C_{\text{opt}}$. The proof is completed. □

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grants 71071123, 60921003, and 71371129 and Program for Changjiang Scholars and Innovative Research Team in University under Grant IRT1173.

References

[1] H.-C. Hwang, S. Y. Chang, and K. Lee, "Parallel machine scheduling under a grade of service provision," *Computers and Operations Research*, vol. 31, no. 12, pp. 2055–2061, 2004.

[2] A. Bar-Noy, A. Freund, and J. Naor, "On-line load balancing in a hierarchical server topology," *SIAM Journal on Computing*, vol. 31, no. 2, pp. 527–549, 2001.

[3] J. Ou, J. Y.-T. Leung, and C.-L. Li, "Scheduling parallel machines with inclusive processing set restrictions," *Naval Research Logistics*, vol. 55, no. 4, pp. 328–338, 2008.

[4] M. Ji and T. C. E. Cheng, "An FPTAS for parallel-machine scheduling under a grade of service provision to minimize makespan," *Information Processing Letters*, vol. 108, no. 4, pp. 171–174, 2008.

[5] G. J. Woeginger, "A comment on parallel-machine scheduling under a grade of service provision to minimize makespan," *Information Processing Letters*, vol. 109, no. 7, pp. 341–342, 2009.

[6] Y. Jiang, "Online scheduling on parallel machines with two GoS levels," *Journal of Combinatorial Optimization*, vol. 16, no. 1, pp. 28–38, 2008.

[7] A. Zhang, Y. Jiang, and Z. Tan, "Online parallel machines scheduling with two hierarchies," *Theoretical Computer Science*, vol. 410, no. 38–40, pp. 3597–3605, 2009.

[8] J. Park, S. Y. Chang, and K. Lee, "Online and semi-online scheduling of two machines under a grade of service provision," *Operations Research Letters*, vol. 34, no. 6, pp. 692–696, 2006.

[9] M. Liu, C. Chu, Y. Xu, and F. Zheng, "Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times," *Journal of Combinatorial Optimization*, vol. 21, no. 1, pp. 138–149, 2011.

[10] M. Liu, Y. Xu, C. Chu, and F. Zheng, "Online scheduling on two uniform machines to minimize the makespan," *Theoretical Computer Science*, vol. 410, no. 21-23, pp. 2099–2109, 2009.

[11] X. Lu and Z. Liu, "Semi-online scheduling problems on two uniform machines under a grade of service provision," *Theoretical Computer Science*, vol. 489-490, pp. 58–66, 2013.

[12] Y. Wu, M. Ji, and Q. Yang, "Optimal semi-online scheduling algorithms on two parallel identical machines under a grade of service provision," *International Journal of Production Economics*, vol. 135, no. 1, pp. 367–371, 2012.

[13] A. Zhang, Y. Jiang, L. Fan, and J. Hu, "Optimal online algorithms on two hierarchical machines with tighly-grouped processing times," *Journal of Combinatorial Optimization*, 2013.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 949726, 5 pages
http://dx.doi.org/10.1155/2014/949726

Research Article

Computing the Expected Cost of an Appointment Schedule for Statistically Identical Customers with Probabilistic Service Times

Dennis C. Dietz

CenturyLink, Inc., Boulder, CO 80301, USA

Correspondence should be addressed to Dennis C. Dietz; dennis.dietz@centurylink.com

Received 29 August 2013; Accepted 13 November 2013; Published 30 January 2014

Academic Editors: E. K. Aydogan, D. Oron, and M. D. Toksari

Copyright © 2014 Dennis C. Dietz. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A cogent method is presented for computing the expected cost of an appointment schedule where customers are statistically identical, the service time distribution has known mean and variance, and customer no-shows occur with time-dependent probability. The approach is computationally efficient and can be easily implemented to evaluate candidate schedules within a schedule optimization algorithm.

1. Introduction

Consider a medical service system where a fixed number of patients are to be scheduled for appointments with a single provider over a finite time horizon [1]. The common service time distribution for every patient is known, as is the time-dependent probability that any patient will fail to show up for the appointment. During the time each arriving patient waits for service, an institutional waiting cost of c_w per hour is incurred. Expected waiting times can be kept small by planning large gaps between appointments, but the gap sizes are constrained by a finite provider availability period. To ensure that all patients are served, the availability period can be extended at an institutional overtime cost of c_o per hour. The objective is to determine a schedule of patient arrival times that will minimize the expected total cost (waiting and overtime) of operating the appointment system.

Similar problems could arise in many other operational contexts. For example, a maritime shipping entity may need to optimally schedule vessel dockings within a fixed window of rented dock time (overtime penalties for docks can be quite severe). A surgical suite manager may need to determine a schedule for procedures that minimizes expected waiting times for surgical teams while avoiding intrusion on a subsequent high-priority commitment. Analogous to medical environments, providers in legal, financial, or other personal service professions that operate on an appointment basis are usually concerned with both server efficiency and client waiting times.

Appointment optimization has received substantial attention in operations research literature, originating with a brief reference by Herne in 1951 in the discussion following Kendall's important article on queueing theory [2]. Numerous attempts followed to obtain reasonable scheduling rules through simulation modeling, beginning with Bailey [3]; relevant surveys are presented in Magerlein and Martin [4], Przasnyski [5], and Ho et al. [6]. Analytic modeling efforts began with steady-state approximation of appointment systems, where statistically identical customers could be scheduled to arrive on a continuous time horizon [7–9]. Transient models and solution methods were later addressed [10–14]. The mathematical complexity of optimization in continuous time led other researchers to constrain arrivals to discrete time points (e.g., a 15-minute lattice) [15, 16]; operationally, this approach was more realistic for most applications. Service time distributions were assumed to be identical and follow exponential, Erlang, or Coxian probability distributions. Vanden Bosch and Dietz [17] generalized the service time model and proposed a heuristic sequencing algorithm for statistically nonidentical customers. Kaandorp

and Koole [18] subsequently derived an exact sequencing and scheduling algorithm for customers having exponentially distributed service times with distinct mean values.

Consistent with the standard practice of scheduling appointments on a lattice of time slots with fixed width Δ, assume that each of N customers is appointed to enter the service queue at one of K discrete times $0, \Delta, 2\Delta, \ldots, (K-1)\Delta$. All customers arrive exactly on time, and the server works under a first-come-first-served (FCFS) discipline whenever one or more customers are in the system. Let a_k be the number of appointments scheduled at time $(k-1)\Delta$, and define a *schedule* as a vector $S = (a_1, a_2, \ldots, a_K)$ such that $\sum_{k=1}^{K} a_k = N$. Any single component of S has a feasible range of 0 to N. For example, a trivial problem where $c_w = 0$ and $c_o > 0$ clearly yields an optimal schedule of $S = (N, 0, 0, \ldots, 0)$.

The first customer should always be scheduled in the first time slot, since a later arrival would waste server time with no improvement in waiting times. The total number of schedules that must be considered is the number of ways in which the remaining $N-1$ appointments can be assigned to K time slots, which can be computed as $\binom{N+K-2}{N-1}$. Hence, a typical problem involving 20 customers and 60 time slots would generate $\binom{78}{19} = 6.71 \times 10^{17}$ candidate schedules. In general, the large number of candidate schedules will prohibit an optimality search by exhaustive enumeration. Assuming that a method exists for determining the total cost $C(S)$ of any schedule, optimality can be efficiently achieved using the algorithm below [1].

(1) Determine a schedule that is assured to have a lower cost than all earlier schedules, S_E, using the following procedure.

  (a) Establish an early incumbent schedule S_E. If no better bound is available, let $S_E = (N, 0, 0, \ldots, 0)$.

  (b) Let m be the largest integer for which the mth arriving customer in S_E is not scheduled at time $(K-1)\Delta$.

  (c) Establish a candidate early schedule S by shifting the arrival of the mth arriving customer in S_E by Δ later, unless this shift causes the order of customer arrivals to change. If all customers but the first are scheduled at $(K-1)\Delta$, stop (recall that the first customer's arrival is fixed at zero).

  (d) If $C(S) < C(S_E)$, let $S_E = S$ and return to step 1(b).

  (e) If $m > 2$, decrement m and return to step 1(c). Otherwise, each customer of the current S_E but the first has shifted without improvement, and S_E is established.

(2) Establish a candidate late schedule S_L by shifting each arrival time by Δ, if feasible. Perform a parallel procedure to step (1) (shifting arrivals by Δ earlier rather than later).

(3) If S_E and S_L differ, the optimal arrival time of each customer is that defined either by S_E or by S_L. Enumerate each of the possible intermediate schedules and evaluate their costs to find the optimum.

Although this algorithm is NP-hard due to the enumeration sometimes required by step (3), S_E and S_L often coincide in practice and seldom differ by more than a few customer arrivals. Computation time is roughly linearly dependent on N and K in the coincidental case, but the algorithm always requires the evaluation of numerous candidate schedules. Computational efficiency is therefore dependent on a cogent and efficient method for computing the expected cost of a specified appointment schedule, which is the contribution offered by this paper.

2. Service Time Model

Suppose that the customer service times are independent, identically distributed random variables with known mean θ and variance σ^2, so the coefficient of variation is $\xi = \sigma/\theta$. For $\xi \le 1$, the service time distribution can be modeled as a mixture of Erlang$(\mu, r-1)$ and Erlang(μ, r) distributions with density function

$$f(t) = \alpha \frac{\mu^{r-1} t^{r-2}}{(r-2)!} e^{-\mu t} + (1-\alpha) \frac{\mu^r t^{r-1}}{(r-1)!} e^{-\mu t}, \quad t \ge 0, \tag{1}$$

where $0 \le \alpha \le 1$. When the parameters r, α, and μ are chosen such that

$$r = \left\lceil \frac{1}{\xi^2} \right\rceil,$$
$$\alpha = \frac{\xi^2 - \left\{ r\left(1+\xi^2\right) - r^2\xi^2 \right\}^{1/2}}{1+\xi^2}, \tag{2}$$
$$\mu = \frac{r-\alpha}{\theta},$$

the distribution will have the required mean and variance (see Tijms [19, page 358]). This model is desirable, since $f(t)$ is always unimodal and is similar in shape to the commonly occurring gamma density function.

For the less typical situation where $\xi > 1$, we can resort to modeling the service time distribution as a mixture of Erlang$(\mu, 1)$ and Erlang(μ, r) distributions with density function

$$f(t) = \alpha\mu e^{-\mu t} + (1-\alpha) \frac{\mu^r t^{r-1}}{(r-1)!} e^{-\mu t}, \quad t \ge 0, \tag{3}$$

where $0 \le \alpha \le 1$. In this case, the required mean and variance can be realized when

$$r = \max\left\{2, \min\left\lceil k : \frac{k^2+4}{4k} \ge \xi^2 \right\rceil\right\},$$
$$\alpha = \frac{2r\xi^2 + r - 2 - \left(r^2 + 4 - 4r\xi^2\right)^{1/2}}{2(r-1)\left(1+\xi^2\right)}, \tag{4}$$
$$\mu = \frac{\alpha + r(1-\alpha)}{\theta}.$$

This approach may be reasonable and useful, but should be applied with caution if distribution characteristics beyond the mean and variance are known. Simulation studies have demonstrated that appointment schedule performance is generally insensitive to higher order moments of the service time distribution [20], but these studies are limited to cases where $\xi \leq 1$.

3. Cost Computation

With the chosen service time model, we can compute the expected total cost of a schedule by exploiting the memoryless property of the exponential distribution. The service times are conceptually comprised of exponentially distributed service phases, each with mean duration $1/\mu$. Hence, the state of the system at any time can be completely described by the number of unfinished phases remaining in the system. Let $t_k = (k-1)\Delta$, $t_k^- = \lim_{\delta \to 0}(t_k - \delta)$, and $t_k^+ = \lim_{\delta \to 0}(t_k + \delta)$. Since each phase completion is an event within a Poisson processs, the probability that s phases of service remain at t_k^- given v remains at t_{k-1}^+ can be written as

$$p(s \mid v) = \begin{cases} \dfrac{e^{-\mu\Delta}(\mu\Delta)^{v-s}}{(v-s)!}, & 0 < s \leq v, \\ 1 - \displaystyle\sum_{m=0}^{v-1} \frac{e^{-\mu\Delta}(\mu\Delta)^{m}}{m!}, & s = 0, \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

for $k = 2, \ldots, K+1$. Now, let $A_k = \sum_{j=1}^{k} a_j$ and let $q_k(s)$ be the probability that s phases remain at t_k^-. For notational convenience, define a binomial operator $b(i, a, \pi) = \binom{a}{i}\pi^i(1-\pi)^{a-i}$, $0 \leq i \leq a$, $0 \leq \pi \leq 1$, and let

$$z = \begin{cases} 1, & \xi \leq 1, \\ r - 1, & \xi > 1. \end{cases} \tag{6}$$

By conditioning on the number of scheduled arrivals that enter the system with fewer than r service phases (i.e., with $r-1$ phases when $\xi \leq 1$ or with a single phase when $\xi > 1$), we have

$$q_2(s) = \sum_{i=0}^{a_1} b(i, a_1, \alpha)\, p(s \mid ra_1 - zi), \quad s = 0, \ldots, ra_1, \tag{7}$$

and, by recursion,

$$q_k(s) = \sum_{i=0}^{a_{k-1}} b(i, a_{k-1}, \alpha) \times \sum_{m=\max(0, s-ra_{k-1}+zi)}^{rA_{k-2}} q_{k-1}(m)\, p(s \mid ra_{k-1} - zi + m), \quad k = 3, \ldots, K+1, \quad s = 0, \ldots, rA_{k-1}. \tag{8}$$

Expected total cost is then given by

$$C = c_w\left(\sum_{k=1}^{K} \frac{a_k(a_k-1)\theta}{2} + \sum_{k=2}^{K}\sum_{s=1}^{rA_{k-1}} q_k(s)\frac{a_k s}{\mu}\right) + c_o\left(\sum_{s=1}^{rA_K} q_{K+1}(s)\frac{s}{\mu}\right). \tag{9}$$

The first summation in (9) is equivalent to $\sum_{k=1}^{K}\sum_{j=1}^{a_k-1} a_j\theta$ and accounts for waiting due to multiple arrivals scheduled in the same time slot.

The method can be easily extended to accommodate probabilistic customer no-shows. Let γ_k be the probability that a customer actually appears for an appointment scheduled in slot k. By conditioning on the number of customers showing up, (7) can be rewritten as

$$q_2(s) = \sum_{j=0}^{a_1} b(j, a_1, \gamma_1) \sum_{i=0}^{j} b(i, j, \alpha)\, p(s \mid rj - zi), \quad s = 0, \ldots, ra_1, \tag{10}$$

and (8) can become

$$q_k(s) = \sum_{j=0}^{a_{k-1}} b(j, a_{k-1}, \gamma_{k-1}) \sum_{i=0}^{j} b(i, j, \alpha) \times \sum_{m=\max(0, s-rj+zi)}^{rA_{k-2}} q_{k-1}(m)\, p(s \mid rj - zi + m), \quad k = 3, \ldots, K+1, \ s = 0, \ldots, rA_{k-1}. \tag{11}$$

Expected total cost is then computed as

$$C = c_w\left(\sum_{k=1}^{K}\sum_{j=2}^{a_k} b(j, a_k, \gamma_k)\frac{j(j-1)\theta}{2} + \sum_{k=2}^{K}\sum_{j=1}^{a_k} b(j, a_k, \gamma_k)\sum_{s=1}^{rA_{k-1}} q_k(s)\frac{js}{\mu}\right) + c_o\left(\sum_{s=1}^{rA_K} q_{K+1}(s)\frac{s}{\mu}\right). \tag{12}$$

The cost computation procedure can be embedded in an appointment optimization algorithm for statistically identical customers. Unlike more sophisticated and generalized approaches [17], the distribution parameters are trivially calculated and cost computation requires no matrix exponentiation. Schedule optimization can thus be easily performed on a personal computer with common software such as a macroenabled spreadsheet. It should be noted that the cost evaluation of an individual candidate schedule may not require complete computation of C; summation and supporting calculations can be terminated as soon as the partial cost of the candidate schedule exceeds the cost of the incumbent.

Table 1: Effect of service time variability.

ξ	r	Optimal schedule	Total cost	Schedules evaluated	Execution time (sec)
0.125	64	(1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0)	1.4072	309	15
0.250	16	(1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0)	2.7861	304	2
0.500	4	(1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0)	6.7935	287	1
1.000	1	(2, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0)	15.9581	253	1
1.500	9	(2, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0)	25.2274	217	1
2.000	16	(2, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0)	32.8035	189	2

4. Implementation and Performance

Now consider a specific appointment scheduling problem where the number of statistically identical customers is $N = 10$, the number of time slots is $K = 16$, the time slot width is $\Delta = 0.5$, the mean service time is $\theta = 0.75$, the service time variance is $\sigma^2 = 0.25$, the show probability is $\gamma_k = 0.95, k = 1, \ldots, 16$, and the cost of server overtime is estimated at ten times the cost of customer waiting (we notionally set $c_w = 1$ and $c_o = 10$, since only relative values affect the optimal schedule). Because $\xi = 0.6667 < 1$, the applicable service time model is given by (1) and the associated parameter values are computed as $r = 3$, $\alpha = 0.5234$, and $\mu = 3.3022$. Imbedding the cost computation method within the optimization algorithm described above yields an optimal schedule of

$$S = (1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0). \quad (13)$$

The associated waiting, overtime, and total costs are 4.8603, 4.9541, and 9.8144, respectively. Optimality is achieved after evaluation of 287 candidate schedules, and complete execution requires less than one second of processing time on a personal computer with a 2.26 GHz processor.

Table 1 illustrates the effect of service time variability by parameterizing on ξ. The table quantifies the intuitive positive relationship between variability in service time and the expected cost of the optimal schedule. The number of schedules evaluated diminishes as ξ increases, although computation times are longer for very high or low values of ξ due to the larger number of exponential phases r in the associated service time models.

To further exercise the modeling approach, we can enlarge the baseline problem (with $\xi = 0.6667$) to schedule $N = 50$ customers into $K = 80$ time slots. The cost of the optimal schedule is 51.8026, which is obtained after evaluating 34,162 schedules. Execution time for this very large problem is 483 seconds, which equates to about 0.0141 seconds per schedule evaluated.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] P. M. Vanden Bosch and D. C. Dietz, "Minimizing expected waiting in a medical appointment system," *IIE Transactions*, vol. 32, no. 9, pp. 841–848, 2000.

[2] D. G. Kendall, "Some problems in the theory of queues," *Journal of the Royal Statistical Society B*, vol. 18, no. 2, pp. 151–185, 1951.

[3] N. T. J. Bailey, "A study of queues and appointment systems in hospital outpatient departments, with special reference to waiting times," *Journal of the Royal Statistical Society B*, vol. 14, no. 2, pp. 185–199, 1952.

[4] J. M. Magerlein and J. B. Martin, "Surgical demand scheduling: a review," *Health Services Research*, vol. 13, no. 4, pp. 418–433, 1978.

[5] Z. H. Przasnyski, "Operating Room Scheduling," *AORN Journal*, vol. 44, no. 1, pp. 67–82, 1986.

[6] C.-J. Ho, H.-S. Lau, and J. Li, "Introducing variable-interval appointment scheduling rules in service systems," *International Journal of Operations and Production Management*, vol. 15, no. 6, pp. 59–68, 1995.

[7] W. R. van Voorhis, "Waiting-line theory as a management tool," *Operations Research*, vol. 4, no. 2, pp. 221–228, 1956.

[8] P. M. Morse, *Queues, Inventories, and Maintenance: the Analysis of Operational Systems with Variable Demand and Supply*, Wiley, New York, NY, USA, 1963.

[9] B. Jansson, "Choosing a good appointment system—a study of queues of the type D/M/1," *Operations Research*, vol. 14, no. 2, pp. 292–312, 1966.

[10] G. R. Grape, "Convergence and cost minimization in queuing systems of the type (D,M,1)," *Försvarets Forskningsanstalt*, vol. 2, no. 1, pp. 1–6, 1968.

[11] B. E. Fries and V. P. Marathe, "Determination of optimal variable-sized multiple-block appointment systems," *Operations Research*, vol. 29, no. 2, pp. 324–345, 1981.

[12] C. D. Pegden and M. Rosenshine, "Scheduling arrivals to queues," *Computers and Operations Research*, vol. 17, no. 4, pp. 343–348, 1990.

[13] P. P. Wang, "Static and dynamic scheduling of customer arrivals to a single-server system," *Naval Research Logistics*, vol. 40, no. 3, pp. 345–360, 1993.

[14] P. P. Wang, "Optimally scheduling N customer arrival times for a single-server system," *Computers and Operations Research*, vol. 24, no. 8, pp. 703–716, 1997.

[15] C.-J. Liao, C. D. Pegden, and M. Rosenshine, "Planning timely arrivals to a stochastic production or service system," *IIE Transactions*, vol. 25, no. 5, pp. 62–73, 1993.

[16] P. M. Vanden Bosch, D. C. Dietz, and J. R. Simeoni, "Scheduling customer arrivals to a stochastic service system," *Naval Research Logistics*, vol. 46, no. 5, pp. 549–559, 1999.

[17] P. M. Vanden Bosch and D. C. Dietz, "Scheduling and sequencing arrivals to an appointment system," *Journal of Service Research*, vol. 4, no. 1, pp. 15–25, 2001.

[18] G. C. Kaandorp and G. Koole, "Optimal outpatient appointment scheduling," *Health Care Management Science*, vol. 10, no. 3, pp. 217–229, 2007.

[19] H. C. Tijms, *Stochastic Models: An Algorithmic Approach*, Wiley, West Sussex, UK, 1994.

[20] C. Ho and H. Lau, "Minimizing total costs in scheduling outpatient appointments," *Management Science*, vol. 38, no. 12, pp. 1750–1764, 1992.