Multiobjective Order Acceptance and Scheduling on Unrelated Parallel Machines with Machine Eligibility Constraints

Bailin Wang and Haifeng Wang
Donlinks School of Economics and Management, University of Science and Technology Beijing, Beijing 100083, China

Correspondence should be addressed to Bailin Wang; wangbl@ustb.edu.cn

Received 27 October 2017; Revised 19 January 2018; Accepted 6 February 2018; Published 7 March 2018

Academic Editor: Eusebio Valero

Copyright © 2018 Bailin Wang and Haifeng Wang. 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 studies the order acceptance and scheduling problem on unrelated parallel machines with machine eligibility constraints. Two objectives are considered to maximize total net profit and minimize the makespan, and the mathematical model of this problem is formulated as multiobjective mixed integer linear programming. Some properties with respect to the objectives are analysed, and then a classic list scheduling (LS) rule named the first available machine rule is extended, and three new LS rules are presented, which focus on the maximization of the net profit, the minimization of the makespan, and the trade-off between the two objectives, respectively. Furthermore, a list-scheduling-based multiobjective parthenogenetic algorithm (LS-MPGA) is presented with parthenogenetic operators and Pareto-ranking and selection method. Computational experiments on randomly generated instances are carried out to assess the effectiveness and efficiency of the four LS rules under the framework of LS-MPGA and discuss their application environments. Results demonstrate that the performance of the LS-MPGA developed for trade-off is superior to the other three algorithms.

1. Introduction

In recent decades, the topic of order acceptance and scheduling (OAS) has attract considerable attention from scheduling researchers and production managers who practice it. The key issue of OAS is to make a joint decision of which orders are accepted (order acceptance decision) and how to schedule them (scheduling decision). Therefore, OAS is essentially different from traditional scheduling problem in which all jobs must be accepted, because the latter is just a special case of it. In OAS, a job has two options, to be accepted or rejected; thus the solution space can be up to $2^n$ times of that of traditional scheduling problem, where $n$ is the number of orders.

Unrelated parallel machine environment is a common workshop where processing times of orders are machine dependent. In traditional OAS on unrelated parallel machines, it is usually assumed that orders are able to be processed on any machines. However, in reality, especially in the assemble lines with multivariety production, one machine is only eligible to process specified orders, which is called the machine eligibility constraint. Recently, some scholars have considered this constraint in the unrelated parallel machine scheduling models (see [1–3]). Moreover, in the assemble line with machine eligibility constraints, processing technologies used in machines are different; thus production costs of one order on different machines are often different and do not depend on the processing times. That is, an order on an eligible machine may have short processing time but high production cost, due to the skilled but not advanced processing technology. Therefore, order assignments on machines affect not only the productivity but also the profit.

This paper studies OAS on unrelated parallel machines with machine eligibility constraints (referred to as OAS-ME). Two objectives are considered. One is to maximize the total net profit with respect to the revenue, tardiness cost, and production cost. The other is to minimize the makespan, which is a classic scheduling criterion with respect to the productivity. Note that the two objectives are conflicting. The makespan is the completion time of the last finished orders; thus the solution with minimum makespan usually does not have the minimum total tardiness, let alone the minimum total
tardiness cost (namely, total weighted tardiness). Moreover, because production cost is independent of the processing time, the minimum makespan does not imply minimal total production cost. Therefore, minimizing the makespan does not mean minimizing the total net profit, and vice versa.

As shown in Section 3.1, OAS-ME is NP-hard; thus this paper presents four list scheduling (LS) rules and then develops a list-scheduling-based multiobjective parthenogenetic algorithm (LS-MPGA). The contributions of this paper are in the following areas: (a) for the OAS-ME, analyse some properties with respect to the objective of total net profit; (b) extend a classic LS rule with the consideration of the net profit, and present three new LS rules according to problem characteristics; (c) propose a LS-based algorithm (named LS-MPGA) with parthenogenetic operators and Pareto-ranking and selection method; (d) suggest application environments of the four LS rules under the framework of LS-MPGA through computational studies. The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 models the problem OAS-ME and analyses some properties. Section 4 proposes four LS rules and Section 5 presents the LS-MPGA. Experimental research in Section 6 inspects performance of these LS rules under the framework of LS-MPGA. Finally, Section 7 concludes the paper.

2. Literature Review

This paper considers an OAS problem on unrelated parallel machines with binary objectives. In the following, we will review the works related to the OAS on parallel machines and multiobjective unrelated parallel machine scheduling algorithms.

By now, extensive studies on OAS have been conducted in the production scheduling literature, and Slotnick [4] provided a comprehensive literature review on it. For parallel machine environment, there are some new studies presented after the review [4]. Wang et al. [5] studied the problem with two identical parallel machines and developed two heuristics and an exact algorithm based on optimal properties and the Lagrangian relaxation technique. Emami et al. [6] considered nonidentical parallel machine environment in which the revenue from an accepted order and processing times are uncertain and developed a Lagrangian relaxation algorithm. Moreover, a series of related papers treat OAS from the perspective of order rejection, and a comprehensive survey was made by Shabtay et al. [7]. For the unrelated parallel machine scheduling with rejection, the new studies after the survey [7] are reviewed as follows. Hsu and Chang [8] studied the problem with deteriorating jobs to minimize the sum of total rejection cost and a scheduling criterion and proved that if the scheduling criterion is either total load or total completion time, the problem is solvable in polynomial time. Lin et al. [9] presented a deterministic 3-approximation algorithm and a randomized 3-approximation algorithm for two unrelated parallel machine scheduling problem with rejection. Jiang and Tan [10] presented a heuristic with worst-case ratio of 2 for an unrelated parallel machine scheduling with rejection and nonsimultaneous machine available time to minimize the sum of the makespan and total rejection cost. Above studies, including literature on unrelated parallel machines in the two survey papers, mostly employed the a priori optimization approach to sum objectives as an aggregated function, which cannot provide Pareto-optimal solutions for trade-off. Moreover, to the best of our knowledge, the machine eligibility constraint and time-independent production cost in our problem have not been considered in OAS previously in the literature.


3. Problem Modelling and Analysis

3.1. Problem Model. OAS-ME can be formally described as follows. Given a set of \( n \) nonpreemptive orders and a set of \( m \) unrelated parallel machines, this problem is to decide which orders should be accepted and how to schedule them. Each accepted order must be processed on one machine, and rejected orders are not allowed to be processed. Each machine can process at most one order at a time. Moreover, each order can only be processed on specific machines (machine eligibility constraints), and processing times and production costs of an order on the eligible machines are different from each other. Two objectives are considered. One is to maximize the total net profit, which is equal to revenues minus the sum of production costs and tardiness costs of accepted orders. The other is to minimize the makespan.

For convenience, following notations are introduced.

(a) Indexes and Sets

- \( j \): order index; \( n \): the number of orders; \( O_j \): the order with index \( j \).
- \( i \): machine index; \( m \): the number of machines; \( M_i \): the machine with index \( i \).
- \( EM_j \): the set of machines which are eligible to process \( O_j \). \( EM_j \subseteq \{ M_1, \ldots, M_m \} \).

(b) Problem Parameters

- \( PC_{ij} \), \( pt_{ij} \): production cost and processing time of \( O_j \) on \( M_i \), respectively.
RV, \( d \), \( TC \): revenue, due date, and unit tardiness cost of \( O \), respectively.

M: a very large positive number.

(c) Decision Variables

\( x \): binary variable for order acceptance decision. If \( x = 1 \), \( O \) is accepted, otherwise, rejected.

\( y \): binary variable representing whether \( O \) is the \( k \)th processed order on \( M \) (\( y = 1 \)) or not (\( y = 0 \)).

\( ct \): nonnegative continuous variable representing the completion time of \( O \).

(d) Objective Parameters

\( PR, T \): the net profit and tardiness of \( O \), respectively.

Mathematical model for the OAS-ME can be formulated as a multiobjective mixed integer linear programming (MILP) as follows.

\[
\begin{align*}
\text{max} \ & \ f_1 = \sum_{j=1}^{n} PR_j - \sum_{j=1}^{n} \left( RV_j - TC_j \times T_j - \sum_{i \in M_j} \left( PC_{ij} \times \sum_{k \in O} y_{ijk} \right) \right) \times x_j \\
\text{min} \ & \ f_2 = C_{max} \\
\text{Subject to:} \ & \ x_j - \sum_{i \in EM_j} \frac{1}{n} \sum_{k=1}^{n} y_{ijk} = 0, \quad \forall j \\
& 1 - \sum_{j=1}^{n} y_{i,j,k+1} + M \sum_{j=1}^{n} y_{ijk} > 0, \quad \forall i, k = 1, \ldots, m - 1 \\
& ct_j - pt_{ij} - ct_j + M \left( 2 - y_{i,j,k+1} - y_{ijk} \right) \geq 0, \quad \forall i, k = 1, \ldots, m - 1 \\
& ct_j - \sum_{i \in EM_j} \sum_{k=1}^{n} y_{ijk} pt_{ij} \geq 0, \quad \forall j \\
& T_j \geq 0, \quad \forall j \\
& T_j \geq ct_j - d_j, \quad \forall j \\
& C_{max} \geq ct_j, \quad \forall j \\
& y_{ijk} = 0, \quad \forall j, k, \forall i \notin EM_j \\
& x_j, y_{ijk} \in \{0, 1\}, \quad \forall j, k \in O, \forall i \in M_j.
\end{align*}
\]
**Theorem 1.** If there is at least one accepted order with a negative net profit, the solution is certainly not optimal.

*Proof.* Denote this solution by \( \delta \) and its total net profit and the makespan by \( \sum PR \) and \( C_{\text{max}} \), respectively. Let the set \( O^- \) include all the orders with negative net profits in \( \delta \). By rejecting the orders in \( O^- \), we can get a solution named \( \delta' \) with two objectives \( \sum PR' \) and \( C_{\text{max}}' \).

First, consider the total net profits of \( \delta \) and \( \delta' \). For the orders in \( O^- \), their net profits in \( \delta' \) (which are equal to zero) are greater than those in \( \delta \) (which are negative). For the accepted orders which are not in \( O^- \), their tardiness in \( \delta' \) must not be greater than those in \( \delta \); thus their net profits in \( \delta' \) are not smaller than those in \( \delta \). Therefore, \( \sum PR' \geq \sum PR \).

Second, consider the makespan of \( \delta \) and \( \delta' \). Rejecting an order will not increase the completion time of other accepted orders; thus \( C_{\text{max}}' \leq C_{\text{max}} \).

Since \( \sum PR' \geq \sum PR \) and \( C_{\text{max}}' \leq C_{\text{max}} \), the solution \( \delta \) is dominated by \( \delta' \); thus \( \delta \) is certainly not optimal. \( \square \)

Based on Theorem 1, we can get following corollaries.

**Corollary 2.** The solution in which all orders are rejected is an extreme optimal solution with \( \sum PR = 0 \) and \( C_{\text{max}} = 0 \).

**Corollary 3.** If an order \( O_j \) satisfies \( RV_j < \min_{i\in EM_j} \{PC_{ij}\} \), then in any optimal solution, it must be rejected; otherwise, if \( RV_j = \min_{i\in EM_j} \{PC_{ij}\} \) and \( O_j \) is accepted in an optimal solution, then there must exist an optimal solution in which \( O_j \) is rejected.

For those orders that will surely be rejected in optimal solutions, they can be rejected in advance; therefore, according to Corollary 3, the following rule is set for preliminary rejection decision.

**Rule 1.** If \( RV_j \leq \min_{i\in EM_j} \{PC_{ij}\} \), then reject \( O_j \).

### 4. List Scheduling Rules

**4.1. Motivation and Algorithm Overview.** As aforementioned, OAS-ME is a joint decision of order acceptance and order scheduling. In the problem model in Section 3.1, the variables \( \{x_{ij} | \forall j\} \) correspond to the order acceptance decision, and the variables \( \{y_{ijk} | \forall i, j, k\} \) and \( \{ct_{j'} | \forall j\} \) correspond to the order assignment subdecision and completion time subdecision of scheduling decision, respectively. According to Constraints (5) and (6), while \( \{x_{ij} | \forall j\} \) and \( \{y_{ijk} | \forall i, j, k\} \) are determined, completion times can be calculated by (13). Therefore, the key to solve OAS-ME is how to find the optimal values of variables \( \{x_{ij} | \forall j\} \) and \( \{y_{ijk} | \forall i, j, k\} \).

OAS-ME is a NP-hard multiobjective combinatorial optimization problem, where it is practically impossible to find an optimal solution in polynomial time. For this kind of problem, constructive heuristics and metaheuristics are effective in producing good solutions in a short time. For parallel machine scheduling problem, list scheduling (LS) is a classical constructive heuristic which provides an assignment rule to schedule orders one by one in a specific order list [17]. In this section, we introduce the order acceptance decision to a classic LS rule in Section 4.2 and present three new LS rules in Sections 4.3–4.5, which can make the order acceptance decision and order assignment decision simultaneously.

Moreover, from Theorem 1 and Corollary 3, we can get the following theorem that gives the upper and lower bounds of the total net profits in optimal solutions. This theorem provides the basis for the design of the scheduling rule named IFH in Section 4.5 (see Section 4.5.1).

**Theorem 4.** Denote the total net profit in an optimal solution as \( \sum PR \); then

\[
0 \leq \sum PR \leq \sum_{j=1}^{n} \left( \max \left\{ 0, RV_j - \min_{i\in EM_j} \{PC_{ij}\} \right\} \right).
\] (12)

*Proof.* For an order \( O_j \) satisfying \( RV_j \geq \min_{i\in EM_j} \{PC_{ij}\} \), ideal condition of its net profit is to schedule \( O_j \) to the machine with the minimal production cost, and this assignment will not cause tardiness, namely, \( PR^*_j \leq RV_j - \min_{i\in EM_j} \{PC_{ij}\} \). According to Theorem 1, the worst condition is to reject \( O_j \); then the net profit is 0, namely, \( PR^*_j \geq 0 \). Therefore, if \( RV_j \geq \min_{i\in EM_j} \{PC_{ij}\} \), then \( 0 \leq PR_j \leq RV_j - \min_{i\in EM_j} \{PC_{ij}\} \); otherwise, according to Corollary 3, \( PR^*_j = 0 \); thus \( 0 \leq PR^*_j \leq \max \{0, RV_j - \min_{i\in EM_j} \{PC_{ij}\}\} \); then Theorem 4 holds. \( \square \)
each decision has been described in detail herein, therefore not further described in the subsections.

For ease of algorithm description, some notations are defined.

\[ AV_i \]: available time of \( M_i \), which is equal to the total processing time of scheduled orders on \( M_i \).

\[ PR_{ij}, c_{ij} \]: the net profit and completion time of \( O_j \) if it is scheduled to \( M_i \), respectively.

\[ C_{max}^j \]: the makespan of scheduled jobs \( \{O_1, O_2, \ldots, O_{j-1}\} \).

\[ C_{max}^j(i) \]: the makespan of partial schedule \( \{O_1, O_2, \ldots, O_j\} \) in which \( O_j \) is scheduled to \( M_i \).

**4.2. Extended FAM (EFAM).** One of the most popular LS rules is the first available machine (FAM) rule, which schedules the next job on the list which is available first [17]. We develop an extended FAM (EFAM) for OASME, in which order rejection decision is complemented and machine eligibility constraint is considered. According to Theorem 1, while scheduling \( O_j \), EFAM first creates a candidate set of machines \( (CS_j) \). \( CS_j \) contains the eligible machines which produce positive net profit for \( O_j \) (14). EFAM then defines Rule 2 to reject \( O_j \) or select a machine for it.

\[ CS_j = \{ i | i \in EM \land PR_{ij} > 0 \} \]  

(14)

**Rule 2.** If \( CS_j = \emptyset \), then reject \( O_j \); otherwise, schedule \( O_j \) to \( M_i \), where \( i^* = \arg \min_{i \in CS_j} \{AV_i \} \).

In EFAM, for one order, creating the candidate set and selecting the first available machine both take \( O(m) \). There are \( n \) orders; thus the complexity of EFAM is \( O(nm) \).

**4.3. Highest Profit First (HPF).** This subsection presents a LS rule named highest profit first (HPF) rule, which prefers selecting the first available machine both take \( O(m) \). HPF first finds a machine \( M_i \), satisfying (15) and then calls Rule 3.

\[ i^* = \arg \max_{M_i \in EM_i} \{PR_{ij} \} \]  

(15)

**Rule 3.** If \( PR_{ij} \leq 0 \), then reject \( O_j \); otherwise, schedule \( O_j \) to \( M_i \).

HPF finds \( i^* \) for an order in \( O(m) \); thus its time complexity is \( O(nm) \).

**4.4. Smallest Makespan First (SMF).** Besides total net profit, OASME has the other objective, which has been described in the subsections. This subsection presents a smallest makespan first (SMF) rule. SMF schedule \( O_j \) to the machine with the minimal completion time of \( O_j \) if \( O_j \) is accepted, and the difficulty to develop SMF is how to determine whether to accept \( O_j \). Here the criterion of EFAM is adopted, namely, if none of the machines can make a positive net profit for the considered order; then reject this order. Therefore, In SMF, the candidate set \( CS_j \) is first created by (14); then Rule 6 is employed. Time complexity of SMF is \( O(nm) \).

***Rule 4.*** If \( CS_j = \emptyset \), then reject \( O_j \); otherwise, schedule \( O_j \) to \( M_i \), where

\[ i^* = \arg \min_{i \in CS_j} \{c_{ij} \} = \arg \min_{i \in CS_j} \{AV_i + pt_i \} \]  

(16)

**4.5. Integrated-Function-Based Rule (IFH).** Above LS rules either extend an existing heuristic, or only focus on one objective, which do not take into account the balance of two objectives. This subsection presents a LS rule to trade-off total net profit and the makespan, which is formalized as Rule 5.

**Rule 5.** If \( CS_j = \emptyset \), then reject \( O_j \); otherwise, schedule \( O_j \) to \( M_i \), where

\[ i^* = \arg \min_{i \in EM_i} \{w_1G^1_{ij} + w_2G^2_{ij} \} \]  

(17)

In (17), \( G^1_{ij} \) and \( G^2_{ij} \) are the functions reflecting the influence of assigning an order on the net profit and the makespan, and \( w_1 \) and \( w_2 \) are the weights of \( G^1_{ij} \) and \( G^2_{ij} \), respectively. \( G^1_{ij} \) and \( G^2_{ij} \) should satisfy two requirements: (a) Minimizing \( G^1_{ij} \) and \( G^2_{ij} \) represent the same effect trend of optimization; (b) \( G^1_{ij} \) and \( G^2_{ij} \) should be on the same order of magnitude. The following gives the ideas to design these functions.

**4.5.1. Function \( G^1_{ij} \) with respect to the Net Profit.** For the net profit of \( O_j \) (\( PR_{ij} \)), according to Theorem 4, Proposition 5 holds.

**Proposition 5.** \( 0 \leq PR_{ij} \leq PR_{ij}^* \), where \( PR_{ij}^* = RV_j - \min_{i \in EM_i} \{PC_{ij} \} \).

Therefore, \( G^1_{ij} \) is defined by (18), in which \( G^1_{ij} \in [0, 1] \), and a small \( G^1_{ij} \) corresponds to a high net profit of \( O_j \).

\[ G^1_{ij} = \frac{PR_{ij}^* - PR_{ij}}{PR_{ij}^*} \]  

\[ = \frac{TC_j \times \max \{ct_j - d_j, 0\} + \{PC_j - \min_{i \in EM_i} \{PC_{ij} \}\}}{RV_j - \min_{i \in EM_i} \{PC_{ij} \}} \]  

(18)

**4.5.2. Function \( G^2_{ij} \) with respect to the Makespan.** For the makespan of partial schedule \( \{O_1, O_2, \ldots, O_j\} \) (\( C_{max}^j \)), Proposition 6 holds.

**Proposition 6.** \( C_{max}^{j-1} \leq C_{max}^j \leq C_{max}^{j-1} + \max_{i \in EM_i} \{pt_i \} \).

**Proof.** If \( O_j \) is rejected, then \( C_{max}^{j-1}(i) = C_{max}^{j-1} \). If \( O_j \) is accepted and scheduled to \( M_i \), then \( C_{max}^j(i) = \max\{ct_j, C_{max}^{j-1} \} \), and the worst condition is to schedule \( O_j \) to the machine with the latest available time and the longest processing time, that is, \( C_{max}^{j-1} \leq C_{max}^j \leq C_{max}^{j-1} + \max_{i \in EM_i} \{pt_i \} \). Therefore, Proposition 6 holds.
Therefore, (19) defines $G_{ij}^2$ which belongs to $[0, 1]$. Clearly, small $G_{ij}^2$ represents a small makespan of $\{O_1, O_2, \ldots, O_j\}$.

\[
G_{ij}^2 = \frac{C_{\max}(i) - C_{\max}^{j-1}}{\max_{i \in EM_j}[pt_{ij}]} = \max \left\{ \frac{C_{\max}(i) - C_{\max}^{j-1} + 0}{\max_{i \in EM_j}[pt_{ij}]} \right\}
\]

4.5.3. Time Complexity of IFH. In IFH, for each order $O_j$, to calculate $G_{ij}^1$ and $G_{ij}^2$, $\min_{i \in EM_j}[PC_{ij}]$ and $\max_{i \in EM_j}[pt_{ij}]$ should be found first, which takes $O(m)$. Therefore, finding $M_j$ for $O_j$ takes $O(m)$, and time complexity of IFH is $O(nm)$.

5. Multiobjective Parthenogenetic Algorithm

List scheduling supposes that the order list has been already specified; thus the difficulty of using this method is how to determine a good order list. Genetic algorithm (GA) is a popular approach to produce near-optimal solutions with flexible encoding scheme and genetic operators, and parthenogenetic algorithm (PGA) is an improved GA which is suitable for combinatorial optimization [18]; thus this paper employs PGA with LS rules to produce order lists and then generate solutions for OAS-ME. This algorithm is named LS-based multiobjective PGA (LS-MPGA), and its key points are elaborated in the following subsections.

5.1. Chromosome Encoding and Initialization. A chromosome in LS-MPGA corresponds to an order list; that is, it contains $n$ genes which are all different, and otherwise this chromosome is invalid.

In population initialization process, LS-MPGA randomly generates $N$ chromosomes (order lists) to maintain diversity, where $N$ is the population size.

5.2. Parthenogenetic Operators. A chromosome in LS-MPGA is a permutation in which all genes should be different. Traditional GA has two genetic operators as crossover and mutation, which both have a difficulty in maintaining validity of a permutation chromosome. Parthenogenetic algorithm (PGA) is an improved GA proposed by Li and Tong [19], in which each chromosome has only one parent [20]. In PGA, gene recombination operators are presented as a parthenogenesis approach instead of crossover, and it can guarantee the validity of the offspring for the chromosomes encoded as permutation. There are three types of gene recombination operators: gene shift operator, gene exchange operator, and gene inverse operator, which are employed in LS-MPGA.

(a) Gene shift operator evolves chromosomes by an insertion process with probability $p_s$. It inserts the gene in position $h_1$ into position $h_2$, where $h_1 \neq h_2$.

(b) Gene exchange operator implements a swap process with probability $p_e$. This operator swaps the genes in positions $k_1$ and $k_2$, where $k_1 \neq k_2$.

(c) Gene inverse operator is a reversion process with probability $p_{inv}$. This operator reverses sequence of genes between position $l_1$ and $l_2$, where $1 \leq l_1 \leq l_2 \leq n$.

Figure 1 illustrates examples of these operators, in which a parent evolves offspring chromosomes by these operators with $h_1 = k_1 = l_1 = 2$ and $h_2 = k_2 = l_2 = 5$.

One issue of parthenogenetic operation is how to compose these genetic operators. Insertion and swap are two of the most widely used neighbourhood structures in scheduling [21, 22]; thus, to keep a pure insertion or swap operation, gene shift operator and gene exchange operator are totally executed at most once in one genetic operation. Furthermore, to maintain the diversity, if neither gene shift operator nor gene exchange operator is executed, gene inverse operator must be applied. These considerations are formalized into following parthenogenetic rules, where $a$, $b$, and $c$ are random numbers within the range of $[0, 1]$.

Rule 6. If $a \leq p_s$, then call gene shift operator().

Rule 7. If $(a > p_s) \land (b \leq p_e)$, then call gene exchange operator().

Rule 8. If $[(a > p_s) \land (b > p_e)] \lor (c \leq p_{inv})$, then call gene-inverse-operator().

5.3. Pareto-Ranking and Selection. OAS-ME is a multiobjective optimization, and Pareto-optimal solutions are practical when considering real-life problems since the final solution of the decision-maker is always a trade-off. There have been many effective multiobjective GAs to approximate the true Pareto points, and a classic one is the fast nondominated sorting genetic algorithm (NSGA-II) proposed by Deb et al. [23]. NSGA-II does not need external archive to store discovered nondominated solutions and has high efficiency and low space complexity [24]; thus its ranking and selection approach is applied to LS-MPGA.

5.4. Procedure of LS-MPGA. LS-MPGA is terminated when the generation reaches the specified maximal generation $T_{\max}$. Main procedure of LS-MPGA is shown in Procedure 1.

The advantage of LS-MPGA can be explained based on three points. The first is the simple chromosome encoding scheme with only $n$ genes due to the utility of LS rule. Second, LS-MPGA employs parthenogenetic operators to effectively avoid reproducing invalid offspring. Last, LS-MPGA can provide a variety of nondominated solutions by Pareto-ranking and selection method.

6. Computational Experiments

This section conducts an experimental study to evaluate the performance of proposed LS rules under the framework of LS-MPGA (referred to as EFAM-MPGA, HPF-MPGA, SMF-MPGA, and IFH-MPGA, respectively). Algorithms in this section are all coded in C# language and implemented on a computer with Intel Core i5-6300U/CPU 2.40 GHz 2.50 GHz and RAM 8.00 GB.

6.1. Comparison Algorithm and Parameter Settings. As mentioned in Section 5.3, NSGA-II is a classic and effective
multiobjective GA [24]; thus we use it as a benchmark algorithm throughout this section. In NSGA-II, the encoding scheme and genetic operators are problem-related, which are described in detail as follows.

(a) Encoding Scheme. OAS-ME is a joint decision problem; thus a chromosome in NSGA-II contains $2n$ genes, which are divided into two subsets $\{a_1,\ldots,a_n\}, \{b_1,\ldots,b_n\}$ to make the order acceptance and order scheduling decisions simultaneously. Subset $\{a_1,\ldots,a_n\}$ corresponds to an order list, where gene $a_i$ is the index of $j$th scheduled order. Subset $\{b_1,\ldots,b_n\}$ determines the accepted orders and their processing machines. Based on the idea proposed by Shabtay et al. [7], if $b_i = m + 1$, then reject order $j$; otherwise, schedule order $j$ to machine $b_i$.

(b) Genetic Operators. The two subsets in a chromosome are restricted by different constraints; thus NSGA-II employs different genetic operators for them. Subset $\{a_1,\ldots,a_n\}$ is a permutation which requires values of all genes to be different; then the one-point crossover and swap mutation are adopted [1]. Subset $\{b_1,\ldots,b_n\}$ is encoded by mutually independent positive integers, and it is evolved by the traditional crossover and mutation operators.

By extensive preliminary experiments, parameters of LS-MPGAs are set as $p_e = p_c = 0.8$ and $p_{inv} = 0.2$. Set $w_1 = 1$ and $w_2 = 2$ for IFH-MPGA. The crossover and mutation probabilities in NSGA-II are 0.9 and 1/n, respectively [23].

In all the LS-MPGAs and NSGA-II, the maximal generation $T_{\text{max}} = 250$, and the size of population $N = 100$ if $n \leq 100$; otherwise, set $N = 200$.

A number of test instances are generated with the following parameter setting. Let $\text{DU}[a,b]$ represent a discrete uniform distribution with a range from $a$ to $b$. Processing time $p_{ij}$ is generated from $\text{DU}[0.5 \times p^*_ij, p^*_ij]$, where $p^*_ij \in \text{DU}[1,99]$. Revenue, production cost, unit tardiness cost, and due date are generated from $\text{DU}[50,550]$, $\text{DU}[50,100]$, $\text{DU}[0.5,1.5]$, and $\text{DU}[50,50 \times n/(m-1)]$, respectively.

Problem sizes of test instances are set as $n = \{20,50,100\}$ and $m = \{2,4,6,8,10\}$; $n = 150$ and $m = \{2,4,6,8,10,15\}$; $n = 200$ and $m = \{2,4,6,8,10,15,20\}$. According to the problem size, this set of experiments are divided into 28 groups, and 50 instances are generated randomly for each group. So there are $28 \times 50 = 1400$ instances in total.

To measure the quality of obtained Pareto front, we adopt the metrics $Y$ and $\Delta$ presented by Deb et al. [23].

Metric $Y$ measures the extent of convergence to reference Pareto front $\text{PF}_{\text{ref}}$ (20). In (20), $n_A$ is the number of obtained Pareto front $\text{PF}_A$ by algorithm $A$, and $d_{i}$ is the Euclidean distance between member $i$ in $\text{PF}_A$ and its nearest member in $\text{PF}_{\text{ref}}$. Small $Y$ means a good convergence.

$$Y = \frac{1}{n_A} \sum_{i=1}^{n_A} d_{i}.$$ (20)
Metric $\Delta$ measures the maintenance of diversity (21). In (21), $d_j$ and $d_f$ are the Euclidean distances between the extreme solutions of $\text{PF}_{\text{ref}}$ and the boundary solutions of $\text{PF}_{\text{A}}$, $d_i$ is the Euclidean distance between consecutive solutions of $\text{PF}_{\text{A}}$, and $\bar{d}$ is the average of all $d_i$. Algorithm with better diversity has a smaller $\Delta$.

$$\Delta = \frac{d_f + d_i + \sum_{i=1}^{n_A-1} |d_i - \bar{d}|}{d_f + d_i + (n_A - 1) \bar{d}}.$$  

(21)

These metrics both contain a reference Pareto front set $\text{PF}_{\text{ref}}$, which normally consists of the true Pareto front. However, true Pareto front is unknown for our problem because of its NP-hardness. To find a $\text{PF}_{\text{ref}}$ that is approximate the true Pareto front, we analyse the respective bound of the two objectives as follows. For OAS-ME, if accepted orders are $n$ orders, then the number of order acceptance solutions is $2^n$. Obviously, bound-points of all order acceptance solutions can construct an outer boundary line of the true Pareto front. However, it is hard to exhaustively list all order acceptance solutions in polynomial time, because for an OAS-ME with $n$ orders, the number of order acceptance solutions is $2^n$, which is an exponential function of the problem size. In order to obtain a reference Pareto front set $\text{PF}_{\text{ref}}$ in a reasonable time, we present a bound-based approach as follows. First, extract the nondominated solutions from all solutions found by the four LS-MPGAs and NSGA-II; then, calculate the bound-points of these nondominated solutions; at last, add the bound-points which are not dominated by the other bound-points to $\text{PF}_{\text{ref}}$. Essentially, the bound-based approach produces a $\text{PF}_{\text{ref}}$ consisting of nondominated bound-points for the solutions found by the four LS-MPGAs and NSGA-II.

6.2. Numerical Results and Analysis. Means (Avg.) and standard deviations (Std.) of the two metrics $Y$ and $\Delta$ for 28 test groups are listed in Tables 1 and 2, respectively.

Results in Table 1 reveal that, in terms of the average value of $Y$, IFH-MPGA is the best, followed by SMF-MPGA, and the four LS-MPGAs are all significantly better than NSGA-II. It indicates the effectiveness of the framework of LS-MPGA. Moreover, the values of $Y$ obtained by IFH-MPGA, SMF-MPGA, and EFAM-MPGA are small. The minimal $Y$ values obtained by IFH-MPGA, SMF-MPGA and EFAM-MPGA are 0.91%, 0.63% and 0.73%, respectively. It implies the good convergence of the three algorithms which can produce the solutions that are close to the bound-points.

It can be observed from Table 1 that, with the increase of $m$, the values of $Y$ by all the algorithms increase obviously, while the growth of those by IFH-MPGA is the slowest. In addition, for the instances with small $m$, such as those with $n = 10$ and $m = 4, n = 50$ and $m = 4, n = 100$ and $m = 6$, and $n = 150, 200$ and $m \leq 10$, values of $Y$ obtained by SMF-MPGA are much better (lower) than those of other LS-MPGAs. For the instances with large $m$, values of $Y$ obtained by IFH-MPGA are significantly better than those of other algorithms.

Experimental results in Table 2 show that the best (smallest) values of $\Delta$ are often obtained by HPF-MPGA, followed by IFH-MPGA. HPF-MPGA produced the best values of $\Delta$ for the instances with large number of orders ($n \geq 100$), and IFH-MPGA for those with small $n$. There is no significant difference between EFAM-MPGA and SMF-MPGA, which are both worse than the above two algorithms but better than NSGA-II on average. Average objective values of the boundary solutions with maximal net profit obtained by the five algorithms are further collected in Table 3, and solution distributions of two randomly generated instances are shown in Figure 2. The following characteristics of these algorithms are observed from these data.

(a) The minimal makespan is often obtained by SMF-MPGA (Figure 2), and the maximal total net profit is often obtained by HPF-MPGA (Table 3 and Figure 2). (b) Through statistics of Table 3, deviations of total net profit obtained by EFAM/SMF/HPF-MPGA/NSGA-II to those by HPF-MPGA are 2.78%, 2.70%, 1.10%, and 4.10%, respectively, and deviations of the makespan are 37.98%, 39.04%, 31.79%, and 9.87%, respectively. It implies that boundary solutions of HPF-MPGA usually have better net profit than those of other algorithms but with much larger makespan. (c) In Figure 2 and Table 3, most of the non-dominated solutions and boundary solutions produced by NSGA-II are dominated by the four LS-MPGAs, again demonstrating that the proposed LS-MPGA framework and LS rules are effective.

To measure the algorithm efficiency, Table 4 collects the average CPU times for all instances and for the instances with three problem sizes $20 \times 2$, $100 \times 10$, and $200 \times 20$. Note that the population size $N$ for the problem size $200 \times 20$ is twice that for $20 \times 2$ and $100 \times 10$; thus these algorithms take much longer time to solve the former instances. Results indicate that, in terms of computational time, TGA, SMF-MPGA, HPS-MPGA, and EFAM-MPGA are very similar and very short, while IFH-MPGA is the longest one. However, for the instances with the largest problem size $200 \times 20$, the least efficient algorithm IFH-MPGA with $T_{\text{max}} = 250$ and $N = 200$ can be terminated in 16 s, which means that the average computational time of one generation is just 0.064 s. It indicates that all these algorithms can produce solutions in a reasonable time.

6.3. Application Environments of These Algorithms. The above experimental results indicate that the performance of LS-MPGAs is much better than that of NSGA-II. According to these results, the application environments of the four LS-MPGA algorithms are suggested as follows.

(a) If a decision-maker prefers the objective of total net profit, HPF-MPGA is a good choice. However, HPF-MPGA
Table 1: Means and standard deviations of Υ × 100 (%).

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>EFAM-MPGA</th>
<th>HPF-MPGA</th>
<th>SMF-MPGA</th>
<th>IFH-MPGA</th>
<th>NSGA-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2</td>
<td>1.84</td>
<td>1.52</td>
<td>4.99</td>
<td>3.87</td>
<td>1.87</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6.35</td>
<td>2.88</td>
<td>18.40</td>
<td>14.80</td>
<td>6.03</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>8.72</td>
<td>5.10</td>
<td>21.06</td>
<td>13.60</td>
<td>9.28</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>11.38</td>
<td>5.67</td>
<td>26.78</td>
<td>18.27</td>
<td>14.08</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>12.02</td>
<td>5.66</td>
<td>30.52</td>
<td>16.68</td>
<td>12.76</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>1.15</td>
<td>0.83</td>
<td>5.78</td>
<td>2.67</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.60</td>
<td>1.65</td>
<td>13.30</td>
<td>6.92</td>
<td>2.56</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>3.95</td>
<td>2.18</td>
<td>18.69</td>
<td>13.64</td>
<td>4.35</td>
</tr>
<tr>
<td>100</td>
<td>8</td>
<td>9.66</td>
<td>5.53</td>
<td>24.64</td>
<td>13.06</td>
<td>8.79</td>
</tr>
<tr>
<td>150</td>
<td>2</td>
<td>0.91</td>
<td>0.61</td>
<td>4.95</td>
<td>2.01</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.47</td>
<td>0.55</td>
<td>6.64</td>
<td>2.93</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>3.74</td>
<td>6.00</td>
<td>8.76</td>
<td>4.54</td>
<td>3.10</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4.49</td>
<td>2.79</td>
<td>8.87</td>
<td>4.03</td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5.60</td>
<td>2.26</td>
<td>11.80</td>
<td>6.56</td>
<td>4.06</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>9.91</td>
<td>10.36</td>
<td>16.38</td>
<td>12.76</td>
<td>7.69</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
<td>0.73</td>
<td>0.46</td>
<td>5.37</td>
<td>1.97</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.35</td>
<td>0.65</td>
<td>6.26</td>
<td>2.83</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2.76</td>
<td>2.27</td>
<td>8.21</td>
<td>4.26</td>
<td>1.80</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4.24</td>
<td>2.25</td>
<td>7.96</td>
<td>4.90</td>
<td>2.14</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5.89</td>
<td>2.87</td>
<td>9.56</td>
<td>4.91</td>
<td>3.12</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>9.91</td>
<td>6.72</td>
<td>9.45</td>
<td>5.76</td>
<td>7.02</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>13.02</td>
<td>10.43</td>
<td>8.18</td>
<td>4.98</td>
<td>9.60</td>
</tr>
</tbody>
</table>

Avg. 5.68 12.48 4.77 4.19 35.18

Figure 2: Solutions of the four algorithms for instances with problem sizes of 100 × 10 and 200 × 10.
has two significant drawbacks: its convergence is the worst and the obtained makespan is much larger than those of other algorithms.

(b) If a decision-maker prefers the objective of the makespan, they can adopt SMF-MPGA or EFAM-MPGA. In particular, SMF-MPGA can achieve the best convergence for the instances with small number of machines. The drawback of the two algorithms is the difficulty to produce a solution with high total net profit.

(c) If a decision-maker would like to find some trade-off solutions, especially for the instances with large number of machines, IFH-MPGA is the best. In terms of convergence, IFH-MPGA is able to produce the lowest value of $\Upsilon$ for most of instances, especially for those with large $m$. In terms of the maintenance of diversity, the value of $\Delta$ is just behind HPF-MPGA. Therefore, the overall performance of IFH-MPGA is superior to the other rules.

### 7. Conclusions

This paper studied the order acceptance and scheduling problem on unrelated parallel machines with machine eligibility constraints (OAS-ME), which is a NP-hard problem. Two objectives are considered as total net profit and the makespan. This paper analyses some properties of the objectives and presents four list scheduling rules named EFAM, HPF, SMF, and IFH. EFAM extends the classic first available machine (FAM) rule. HPF and SMF are the heuristics mainly designed for one objective as total net profit and the makespan, respectively. IFH presents an integrated function to balance the two objectives. Time complexities of these rules are all $O(\text{n}\cdot m)$. Based on the list scheduling method, we develop a multiobjective parthenogenetic algorithm (LS-MPGA) with parthenogenetic operators and Pareto-ranking and selection method. Computational experiments compared the performances of the four LS rules under the framework of LS-MPGA, analysed their strengths and shortcomings and suggested their application environments.

### Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.
Table 3: Average values of objectives in boundary solutions with maximal net profit.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>EFAM-MPGA</th>
<th>HPF-MPGA</th>
<th>SMF-MPGA</th>
<th>IFH-MPGA</th>
<th>NSGA-II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \sum PR )</td>
<td>( C_{\text{max}} )</td>
<td>( \sum PR )</td>
<td>( C_{\text{max}} )</td>
<td>( \sum PR )</td>
</tr>
<tr>
<td>2</td>
<td>4753</td>
<td>497</td>
<td>4760</td>
<td>545</td>
<td>4752</td>
<td>496</td>
</tr>
<tr>
<td>4</td>
<td>4744</td>
<td>293</td>
<td>4764</td>
<td>369</td>
<td>4737</td>
<td>282</td>
</tr>
<tr>
<td>6</td>
<td>4613</td>
<td>220</td>
<td>4649</td>
<td>297</td>
<td>4602</td>
<td>216</td>
</tr>
<tr>
<td>8</td>
<td>4594</td>
<td>168</td>
<td>4642</td>
<td>246</td>
<td>4572</td>
<td>161</td>
</tr>
<tr>
<td>10</td>
<td>4785</td>
<td>156</td>
<td>4835</td>
<td>219</td>
<td>4757</td>
<td>146</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1153</td>
<td>1145</td>
<td>11564</td>
<td>1339</td>
<td>11531</td>
<td>1147</td>
</tr>
<tr>
<td>4</td>
<td>1131</td>
<td>591</td>
<td>11450</td>
<td>899</td>
<td>11315</td>
<td>588</td>
</tr>
<tr>
<td>6</td>
<td>1158</td>
<td>422</td>
<td>11814</td>
<td>711</td>
<td>11592</td>
<td>415</td>
</tr>
<tr>
<td>8</td>
<td>1153</td>
<td>348</td>
<td>11694</td>
<td>574</td>
<td>11457</td>
<td>340</td>
</tr>
<tr>
<td>10</td>
<td>1158</td>
<td>299</td>
<td>11854</td>
<td>524</td>
<td>11567</td>
<td>291</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>23094</td>
<td>1992</td>
<td>22672</td>
<td>2628</td>
<td>22082</td>
<td>1977</td>
</tr>
<tr>
<td>4</td>
<td>22881</td>
<td>1053</td>
<td>23442</td>
<td>1861</td>
<td>22892</td>
<td>1061</td>
</tr>
<tr>
<td>6</td>
<td>22915</td>
<td>736</td>
<td>23644</td>
<td>1397</td>
<td>22930</td>
<td>726</td>
</tr>
<tr>
<td>8</td>
<td>23031</td>
<td>570</td>
<td>23915</td>
<td>1082</td>
<td>23094</td>
<td>562</td>
</tr>
<tr>
<td>10</td>
<td>23313</td>
<td>473</td>
<td>24255</td>
<td>961</td>
<td>23350</td>
<td>466</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>34764</td>
<td>2910</td>
<td>35141</td>
<td>4028</td>
<td>34758</td>
<td>2899</td>
</tr>
<tr>
<td>4</td>
<td>34032</td>
<td>1465</td>
<td>35052</td>
<td>2571</td>
<td>34061</td>
<td>1462</td>
</tr>
<tr>
<td>6</td>
<td>34419</td>
<td>1072</td>
<td>35732</td>
<td>1929</td>
<td>34510</td>
<td>1062</td>
</tr>
<tr>
<td>8</td>
<td>34356</td>
<td>794</td>
<td>35908</td>
<td>1471</td>
<td>34457</td>
<td>785</td>
</tr>
<tr>
<td>10</td>
<td>34224</td>
<td>684</td>
<td>35856</td>
<td>1292</td>
<td>34273</td>
<td>660</td>
</tr>
<tr>
<td>15</td>
<td>33892</td>
<td>487</td>
<td>35761</td>
<td>990</td>
<td>34050</td>
<td>474</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>45615</td>
<td>3781</td>
<td>46166</td>
<td>5303</td>
<td>45612</td>
<td>3761</td>
</tr>
<tr>
<td>4</td>
<td>45333</td>
<td>1921</td>
<td>46847</td>
<td>3226</td>
<td>45388</td>
<td>1918</td>
</tr>
<tr>
<td>6</td>
<td>45051</td>
<td>1364</td>
<td>46977</td>
<td>2346</td>
<td>45131</td>
<td>1344</td>
</tr>
<tr>
<td>8</td>
<td>45376</td>
<td>1031</td>
<td>47679</td>
<td>1884</td>
<td>45527</td>
<td>1015</td>
</tr>
<tr>
<td>10</td>
<td>45133</td>
<td>861</td>
<td>47666</td>
<td>1619</td>
<td>45308</td>
<td>845</td>
</tr>
<tr>
<td>15</td>
<td>44955</td>
<td>609</td>
<td>47725</td>
<td>1184</td>
<td>45147</td>
<td>591</td>
</tr>
<tr>
<td>20</td>
<td>44872</td>
<td>490</td>
<td>47921</td>
<td>1023</td>
<td>45138</td>
<td>469</td>
</tr>
</tbody>
</table>

Table 4: CPU time (in seconds) of the four algorithms.

<table>
<thead>
<tr>
<th>( n \times m )</th>
<th>( N )</th>
<th>EFAM-MPGA</th>
<th>HPF-MPGA</th>
<th>SMF-MPGA</th>
<th>IFH-MPGA</th>
<th>NSGA-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 × 2</td>
<td>100</td>
<td>0.87</td>
<td>0.86</td>
<td>0.84</td>
<td>0.89</td>
<td>0.88</td>
</tr>
<tr>
<td>100 × 10</td>
<td>100</td>
<td>2.15</td>
<td>2.08</td>
<td>1.99</td>
<td>2.96</td>
<td>1.95</td>
</tr>
<tr>
<td>200 × 20</td>
<td>200</td>
<td>9.74</td>
<td>9.78</td>
<td>9.25</td>
<td>15.52</td>
<td>8.07</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>3.92</td>
<td>3.73</td>
<td>3.63</td>
<td>5.04</td>
<td>3.62</td>
</tr>
</tbody>
</table>

Acknowledgments

This work was supported by the Humanity and Social Science Youth Foundation of Ministry of Education of China (no. 17YJC630143), the National Natural Science Foundation of China (nos. 71701016 and 71471015), the Beijing Natural Science Foundation (no. 9174038), and the Fundamental Research Funds for the Central Universities (no. FRF-BD-16-006A).

References


