Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 3124182, 10 pages
https://doi.org/10.1155/2018/3124182

Research Article

Scheduling Batch Processing Machine Using Max–Min Ant System Algorithm Improved by a Local Search Method

XiaoLin Li[1] **and Yu Wang**[2]

[1]*School of Mines, China University of Mining and Technology, Xuzhou 221116, China*
[2]*School of Management, University of Science and Technology of China, Hefei 230026, China*

Correspondence should be addressed to Yu Wang; YuWangUSTC@126.com

Received 4 September 2017; Revised 13 December 2017; Accepted 1 January 2018; Published 28 January 2018

Academic Editor: Fiorenzo A. Fazzolari

The problem of minimizing the makespan on single batch processing machine is studied in this paper. Both job sizes and processing time are nonidentical and the processing time of each batch is determined by the job with the longest processing time in the batch. Max–Min Ant System (MMAS) algorithm is developed to solve the problem. A local search method MJE (Multiple Jobs Exchange) is proposed to improve the performance of the algorithm by adjusting jobs between batches. Preliminary experiment is conducted to determine the parameters of MMAS. The performance of the proposed MMAS algorithm is compared with CPLEX as well as several other algorithms including ant cycle (AC) algorithm, genetic algorithm (GA), and two heuristics, First Fit Longest Processing Time (FFLPT) and Best Fit Longest Processing Time (BFLPT), through numerical experiment. The experiment results show that MMAS outperformed others especially for large population size.

1. Introduction

Scheduling of batch processing machine is a typical combinatorial optimization problem. Different from traditional scheduling problems, batch processing machine can process several jobs simultaneously as a batch. Scheduling batch processing machine is usually encountered in manufacturing industry such as heat treatment in metal industry and environment stress-screening in integrated circuit production. As these operations often tends to be the bottleneck of manufacturing sequence, scheduling batch processing machine will effectively promote the completion time of jobs.

The problem was first proposed by Ikura and Gimple [1] who studied the problem with identical job sizes and constant batch processing time and machine capacity was defined by the number of jobs processed simultaneously. Considering the burning operation in semiconductor manufacturing, Lee et al. [2] presented an efficient dynamic programming based algorithms for minimizing a number of different performance measures. The same problem was studied by Sung and Choung [3] who proposed branch and bound algorithm and several heuristics to minimize the objective of makespan.

The problem is much more complicated when nonidentical job sizes are considered. Uzsoy [4] studied the problem of scheduling single batch processing machine under the objectives of minimizing makespan ($C_{\max}$) and the total processing time ($\sum C_i$) with nonidentical job sizes. Both problems were proved to be NP-hard and several heuristics were proposed including First Fit Longest Processing Time (FFLPT), first fit shortest processing time (FFSPT) et al. Dupont and Jolai Ghazvini [5] proposed two effective heuristics' successive knapsack problem (SKP) and best fit longest processing time (BFLPT) and later one outperformed FFLPT. To solve the problem optimally, exact algorithm like branch and bound was proposed by Dupont and Dhaenens-Flipo [6], they present some dominance properties for a general enumeration scheme for the makespan criterion. Enumeration scheme [7] was also developed combined with existing heuristics to solve large scale problems.

Since the batch processing machine scheduling problem is NP-hard [4], various metaheuristic algorithms have been developed to solve the problem. Melouk et al. [8] studied the problem using Simulated Annealing (SA), and random instances were generated to evaluate the effectiveness of the

algorithm. The same problem was considered by Damodaran et al. [9] with genetic algorithm (GA). The experiment showed that GA outperformed SA in run time and solution quality. Husseinzadeh Kashan et al. [10] proposed the grouping version of the particle swarm optimization (PSO) algorithm and the application of which was made to the single batch-machine scheduling problem. A GRASP approach developed by Damodaran et al. [11] was used to minimize the makespan of a capacitated batch processing machine and the experimental study concluded that GRASP outperformed other solution approaches. For the problems considering multimachines, Zhou et al. [12] proposed an effective differential evolution-based hybrid algorithm to minimize makespan on uniform parallel batch processing machines and the algorithm was evaluated by comparing with a random keys genetic algorithm (RKGA) and a particle swarm optimization (PSO) algorithm. Similar problem was studied by Jiang et al. [13] considering batch transportation. A hybrid algorithm combining the merits of discrete particle swarm optimization (DPSO) and genetic algorithm (GA) is proposed to solve this problem. The performance of the proposed algorithms was improved by using a local search strategy as well. All of these studies show the effectiveness of solving batch processing machines problems by using metaheuristic algorithms.

The studies reviewed above mainly solve the problem by sequencing the jobs into job list and then grouping the jobs into batches. Different from the existing studies, a metaheuristic algorithm MMAS (Max–Min Ant System) was designed in a constructive way by combining these two stages of decisions together. That is to say, the batches are constructed directly without considering job sequences and then the batches process on a batch processing machine. In the process of batch construction, jobs to be added to the existing batches can be selected elaborately by considering batch utilization and batch processing time. To improve the global searching ability of MMAS, local search method based on multiple jobs iterative exchange was proposed.

The remaining part of this paper is organized as follows. The mathematic model of the problem studied in this paper is presented in Section 2. In Section 3, we show the detailed MMAS algorithm used to solve the problem under the study. The parameters tuning and the numeric experimentation are given in Section 4. The paper is finally concluded in Section 5.

2. Mathematic Model

The problem of scheduling single batch processing machine is studied and the objective is to minimize makespan. Batch processing machine can process several jobs as a batch and all the jobs in that batch have the same start and completion time. The process cannot be interrupted once the process begins and no jobs can be added or removed from the machine until all jobs have been finished. Batch processing time is determined by the job with longest processing time in the batch. Jobs are all available at the time of zero.

Symbols and notations used in this paper are listed as follows:

(1) There are n jobs $J = \{1, 2, \ldots, n\}$ to be processed and each job j has nonidentical processing time p_j and size s_j.

(2) The capacity of the machine is assumed to be C and each job j has $s_j \leqslant C$. Job list J will be scheduled into batches $b \in B$ before they are processed where B denotes a batch list, that is, a feasible solution, and $|B|$ means the number of batches in B. Processing time of each batch b equals $P^b = \max\{p_j \mid j \in b\}$.

(3) The objective is to minimize makespan ($C_{\max}$) which is equal to the total batch processing time in a solution B.

Base on the assumptions and notations given above, we can get the following mathematic model of the problem.

$$\min \quad C_{\max} = \sum_{b \in B} P^b \tag{1}$$

$$\text{s.t.} \quad \sum_{b \in B,} x_{jb} = 1 \quad \forall j \in J \tag{2}$$

$$\sum_{j=1}^{n} s_j \cdot x_{jb} \leq C \quad \forall b \in B \tag{3}$$

$$P^b \geq p_j x_{jb} \quad \forall j \in J,\ b \in B \tag{4}$$

$$x_{jb} \in \{0, 1\} \quad \forall j \in J,\ b \in B \tag{5}$$

$$\left\lceil \frac{\sum_{j=1}^{n} s_j}{C} \right\rceil \leq |B| \leq n \quad |B| \in Z^+. \tag{6}$$

Objective (1) is to minimize the makespan. As only one processing machine is considered, the makespan is equal to the total completion time of all batches formed. Constraint (2) ensures that each job j can be assigned exactly to one batch. Constraint (3) guarantees that total size of jobs in a batch does not exceed the machine capacity C. Constraint (4) explains that the batch processing time is determined by the job with longest processing time in that batch. Constraint (5) denotes the binary restriction of variable x_{jb} which is equal to 1 if job j is assigned to batch B and $x_{jb} = 0$ otherwise. Constraint (6) gives the upper and lower bound of the number of batches in a feasible solution B. The lower bound is calculated when assuming jobs can be processed partially across the batches [4]. And the upper bound is generated when each batch accommodates only one job.

3. Max–Min Ant System

MMAS [14] is one of the most successful variants in the framework of ant colony optimization (ACO) [15, 16] which have been applied to many combinatorial optimization problems such as scheduling problems [17], traffic assignment problems [18], and travelling salesman problems [15]. In MMAS, pheromone trail limits interval $[\tau_{\min}, \tau_{\max}]$ is used to prevent premature convergence and to exploit the best solutions found during the solution search. The performance

of the algorithm is significantly affected by the values of the parameters; thus parameters tuning is performed to optimize algorithm performance. A local search method is also developed to enhance the search ability of MMAS.

MMAS is a constructive metaheuristic algorithm which is able to build a solution step by step. It can be adapted to various combinatory optimization problems with a few modifications. Given a list of jobs, MMAS will group the jobs into batches by adding jobs to the existing or new batches one at a time. The sequence of selecting jobs to be constructed into batches depends on the state transition probability calculated according to density of pheromone trail and heuristic information of each solution element. A solution is generated once all jobs are arranged into a batch.

3.1. Pheromone Trails. When solving the problem of TSP (travelling salesman problem) by using ant colony optimization algorithms [15], the pheromone trails τ_{ij} are defined by the expectation of choosing city j from city i, that is, the amount of pheromone of the arc(i, j). The city with higher density of pheromone trail will be selected with a higher possibility. However, in the problem of scheduling batch processing machine, each solution is a set of batches and the sequence of jobs in a batch does not affect the batch processing time; thus the pheromone trails imply the expectation of arranging a job into a batch. In this study, we measure the expectation of adding a job to a batch by using the average pheromone trails between the job j and each job in batch b as follows.

$$\vartheta_{jb} = \frac{1}{|b|} \sum_{k \in b} \tau_{jk}, \tag{7}$$

where τ_{jk} means the pheromone trail between job j and the existing job k in batch b. ϑ_{jb} denotes the expectation of adding the job j to the current batch b. $|b|$ denotes the number of jobs in a batch b. And this expectation will be used as pheromone trails in the calculation of state transition probability.

3.2. Heuristic Information. As the objective $C_{\max}$ equals the total processing time of all batches formed, the quality of a solution is affected by both the number of batches and the processing times of each batch in the solution. Thus, two kinds of heuristic information are considered in this study, that is, the utilization of machine capacity and efficiency of batch processing time.

Usually, solutions with smaller number of batches generate better results. To reduce the unoccupied capacity of each batch, we add job j to batch b with the most feasible capacity in priority according to FFD (first fit decreasing) algorithm in bin packing problem [19]. The heuristic to add a feasible job j to batch b is defined as follows:

$$\mu_{jb} = s_j. \tag{8}$$

The processing time of a batch is determined by the job with the longest processing time in the batch. Obviously, jobs with similar processing times should be batched together to increase the efficiency of batch processing time. Thus, we give another heuristic information for adding job j to batch b as follows:

$$\eta_{jb} = \frac{1}{1 + \left| P^b - p_j \right|}. \tag{9}$$

3.3. Solution Construction. For each ant a, a solution is constructed by selecting an unscheduled job j and adding it to an existing batch b according to a state transition probability P^a_{bj}. If no existing batches can accommodate the job j, a new empty batch will be created and accommodate it. A solution will be generated when all jobs are scheduled into a batch. Since a solution construction depends on the sequence of jobs chosen, solution quality is significantly affected by state transition probability. The probability is determined by the pheromone trails and heuristic information between the current batch and job to be scheduled. The state transition probability is defined as follows:

$$P^a_{bj} = \begin{cases} \dfrac{\vartheta^\alpha_{bj} \cdot \eta^\beta_{bj} \cdot \mu^\gamma_{bj}}{\sum_{j \in V} \vartheta^\alpha_{bj} \cdot \eta^\beta_{bj} \cdot \mu^\gamma_{bj}}, & j \in V \\ 0, & \text{Else,} \end{cases} \tag{10}$$

where V denotes the feasible jobs that can be added to the current batch b such that the machine capacity is not violated. α, β, and γ show the relative importance of pheromone trails and two kinds of heuristic information. For each ant a, a feasible job j in V will be selected with probability and added to the current batch or a new batch until all jobs are scheduled.

3.4. Update of Pheromone Trails. The density of pheromone trails is an important factor in the process of solution construction, as it indicates the quality of solution components. When all ants build its feasible solution, the pheromone trails on every solution component will be updated through pheromone depositing and evaporating. After each iteration, the pheromone trails of each solution component will be decreased with the evaporation rate ρ while solution component of the iterative best solution or the global best solution will be increased by a quantity $\Delta\tau_{jk}$. The pheromone update for each ant a at tth iteration between the solution components of job j and job k is performed according to (11) as follows.

$$\tau_{jk}(t+1) = \rho\tau_{jk}(t) + \Delta\tau_{jk} \tag{11}$$

$$\Delta\tau_{jk} = \begin{cases} \dfrac{1}{C^a_{\max}}, & \text{if job } j \text{ and } k \text{ are of the same batch} \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

where ρ $(0 \le \rho < 1)$ controls the pheromone evaporation rate. $C^a_{\max}$ denotes the solution makespan get by ant a.

The pheromone trails are limited in the interval of $[\tau_{\min}, \tau_{\max}]$ in MMAS algorithm. We set $\tau_{\max} = 1/[\rho \times C^*_{\max}]$ and $\tau_{\min} = \tau_{\max}/2n$ in this study according to Stützle and Hoos [14]. $C^*_{\max}$ is the global best makespan that ants have found.

3.5. Local Search Algorithm. Solution qualities can be effectively improved when local search methods are applied in metaheuristics [20]. That is because greedy strategies are usually adopted in local search methods and local optimal can be easily found in the neighborhood of a given solution. Metaheuristics' ability in global optimal searching can be enhanced by combining their advantages in exploring solution space with local search methods.

As batch processing time is determined by the job with the longest processing time in the batch, jobs with smaller processing time have no effect on batch processing time. Local search method can be applied to decrease batch processing time by batching jobs with similar processing time together.

Definition 1. The job i with longest processing time in batch b is called dominant job d^b. The total processing time of jobs without dominant job is denoted as dominated job processing time, noted as DT.

Proposition 2. *The makespan will be minimized by maximizing DT.*

Proof. We have $C_{\max} = \sum_{b\in B} \max\{p_j \mid j \in b\}$ for a given solution B. According to Definition 1, $DT = \sum_{b\in B}(\sum_{j\in b} p_j - \max\{p_j \mid j \in b\}) = \sum_{b\in B}\sum_{j\in b} p_j - \sum_{b\in B}\max\{p_j \mid j \in b\}$; thus we have $C_{\max} = \sum_{b\in B}\sum_{j\in b} p_j - DT$. As the total job processing time $\sum_{b\in B}\sum_{j\in b} p_j$ is constant for a given instance, the makespan will be minimized when DT is maximized. □

Definition 3. For each batch b, the sum of its remaining capacity and the size of dominant job d^b is called exchangeable capacity EC^b; we denote $\mathrm{EC}^b = C - \sum_{i\in b} s_i + s_{j=\mathrm{argmax}\{p_j \mid j\in b\}}$.

Proposition 4. *Given $p^b \geqslant p^q$, larger DT will be obtained by exchange d^b of batch b with m jobs in batch q where $\sum_{i=1}^{m} s_i \leq \mathrm{EC}^q$ and $\max\{p_i \mid i = 1,\ldots,m\} \leq \max\{p_j \mid j \in q\}$, while the $C_{\max}$ does not increase.*

Proof. Given two batches b and q, $p^b \geqslant p^q$, jobs in batch b can be divided into three sets $\{d^b\}$, m jobs M, and other jobs O, where $\sum_{i=1}^{m} s_i \leq \mathrm{EC}^q$ and $\max\{p_i \mid i = 1,\ldots,m\} \leq \max\{p_j \mid j \in q\}$. Jobs in batch q can be divided into sets $\{d^q\}$ and other jobs O'. Therefore, we have $C_{\max} = p_{d^b} + p_{d^q}$. After the exchange is applied to batches, we have $C'_{\max} = p_{d^b} + p'_{d^q}$, where $p'_{d^q} = \max\{p_i \mid i \in M \cup O'\}$. As $p_{d^q} \geq \{p_i \mid i \in M \cup O'\}$, the relation $C'_{\max} \leq C_{\max}$ is satisfied. According to Proposition 2, there is $C_{\max} = \sum_{j\in b\cup q} p_j - DT$; a minor $C_{\max}$ indicates a larger DT. Proposition 4 holds. □

According to Proposition 4, multiple jobs can be exchanged iteratively between batches to decrease batch processing time. For a given solution S, the number of batches is limited by the total number of jobs where each job is grouped as one batch. The detailed procedure of the proposed local search algorithm MJE (Multiple Jobs Exchange) is listed as follows:

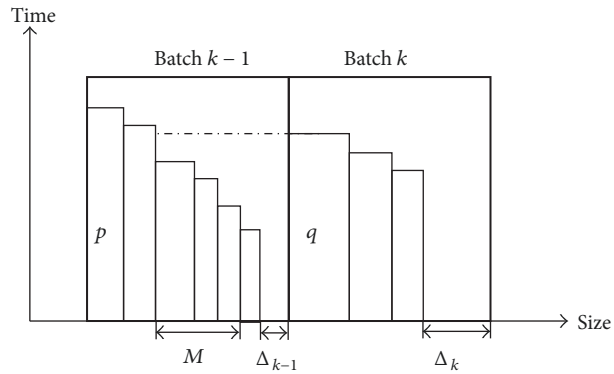

Figure 1: Notations description of Algorithm MJE.

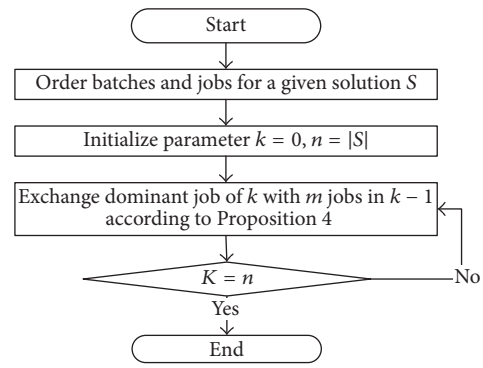

Figure 2: Flow chart of algorithm MJE.

Algorithm MJE (Multiple Jobs Exchange)

Step 1. For the iterative best solution S, arrange the batches in decreasing order of their processing time and order jobs of each batch in decreasing order of job size.

Step 2. Initialize parameter $k = 0$, set $n = |S|$, Δ_k is the remaining capacity of batch k, and p and q denote the dominated job of batch $k-1$ and batch k, respectively.

Step 3. If $k = n$, exit; else $k = k + 1$.

Step 4. Exchange the job q of batch k with m $(m > 0)$ jobs M in batch $k-1$ if $\max\{p_i \mid i \in M\} \leq p_q$, $\sum_{i\in M} s_i \leq s_q + \Delta_k$ and $s_q \leq \sum_{i\in M} s_i + \Delta_{k-1}$; go to Step 3.

Corresponding notations are illustrated in Figure 1.
The flow chart of algorithm MJE is provided as in Figure 2.

3.6. Algorithm Framework of MMAS with Local Search. According to basic framework of ant algorithms [16], the framework of MMAS to solve the problem under study is given as follows.

Algorithm MMAS

Step 1. Initialize parameters in MMAS including α, β, γ, and $\tau(0)$.

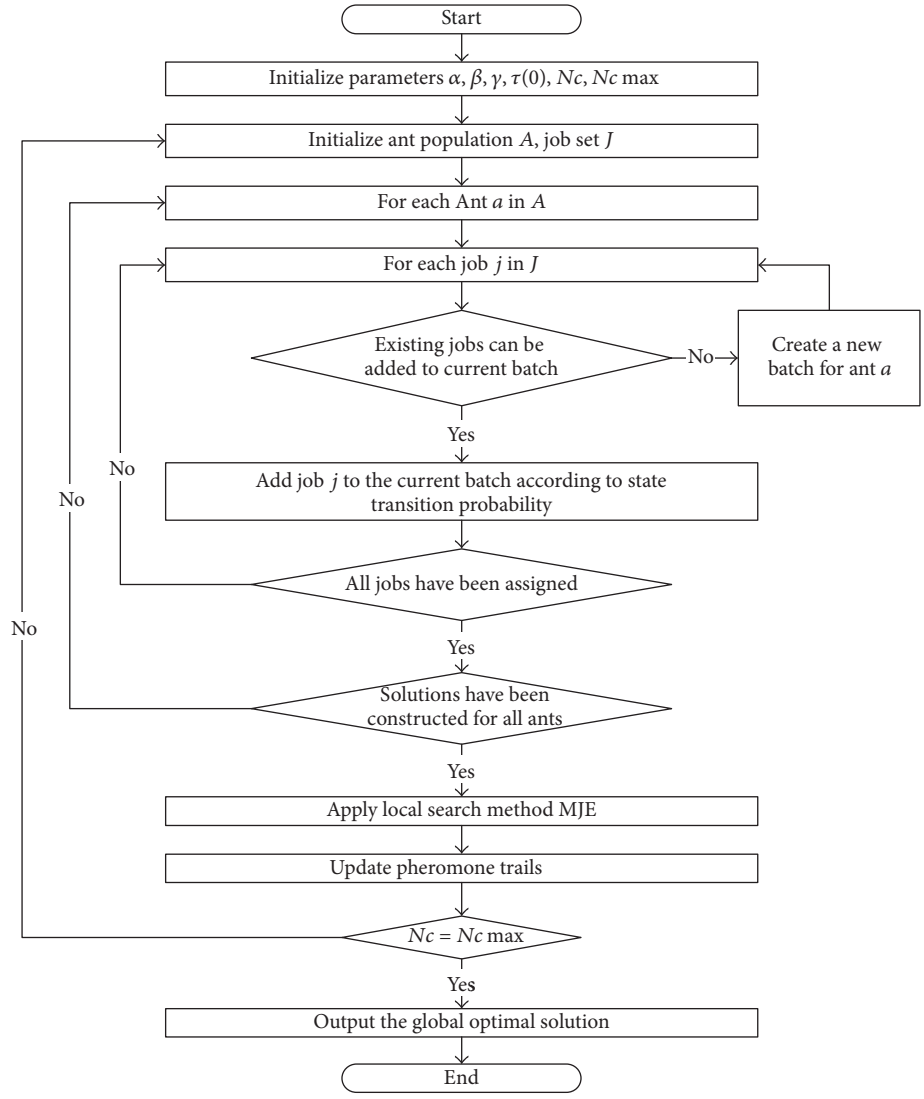

Figure 3: Flow chart of algorithm MMAS.

Step 2. Initialize ant population.

Step 3. Select a unscheduled job for each ant a and add the job to a new batch.

Step 4. Arrange the next unscheduled feasible job with the maximum state transition probability calculated by (10) into the current batch.

Step 5. If there are existing jobs that can be added to the current batch, then go to Step 4.

Step 6. If there are jobs unscheduled, then go to Step 3.

Step 7. Apply local search method MJE to iteration solutions.

Step 8. Update pheromone trails according to formula (11).

Step 9. If the termination condition is satisfied, then output the solution. Else go to Step 2.

To illustrate the procedure of MMAS more logically, the flow chart of algorithm MMAS is provided as in Figure 3.

4. Experimentation

4.1. Experimental Design. Random instances were generated to verify the effectiveness of the proposed algorithm. Three factors were considered in the numeric experiment including number of jobs J, job processing time P, and job size S. 24 ($4 * 2 * 3$) problem categories were generated and denoted in the form of $JiPjSk$ ($i = 1, 2, 3, 4$; $j = 1, 2$; $k = 1, 2, 3$). Factors and levels are shown in Table 1. The machine capacity is assumed to be 10 for all problem instances.

For example, $J1P2S3$ represented the category of instances with 10 jobs, jobs' processing time were randomly generated from a discrete uniform $[1, 20]$ distribution, and jobs' sizes were randomly generated from a discrete uniform $[4, 8]$ distribution.

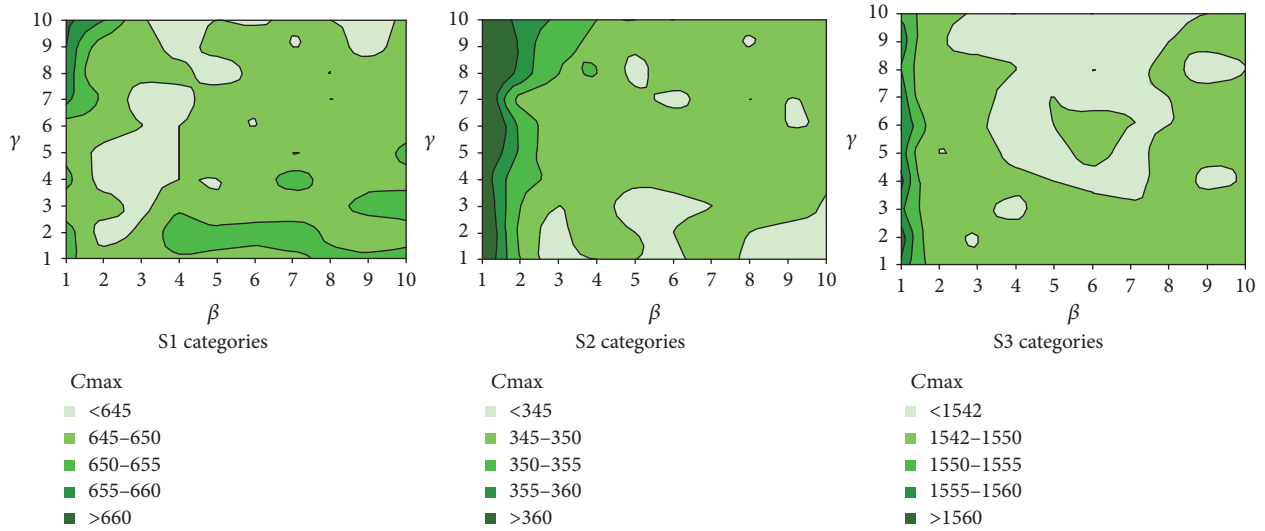

Figure 4: Contour lines of C_{max} influenced by parameters α, β, and γ for different problem categories.

Table 1: Factors and levels.

Factors	Levels
J	$J1 = 10$; $J2 = 20$; $J3 = 50$; $J4 = 100$
P	$P1$: $U[1, 10]$; $P2$: $U[1, 20]$
S	$S1$: $U[1, 10]$; $S2$: $U[2, 4]$; $S3$: $U[4, 8]$

U means data are generated from discrete uniform distribution.

Table 2: Values for parameters.

Run code	β	γ
$S1$	3	5
$S2$	6	3
$S3$	6	8

4.2. Parameter Tuning. As each ant will build feasible solutions with probability, a large population of ants will enhance the algorithm's ability in exploring solution space while more times of iteration help to make better use of pheromone trails and heuristic information. However, the algorithm is much more time consuming with larger number of ants and iterations. Preliminary tests were done and the results showed that to increase the number of iteration yields better results in a given run time. That is easy to be understood as the algorithm inclines to perform randomly search in the initial stages and more empirical results are considered along with the increasing of iterations. To obtain a tradeoff between solution quality and time cost, ant population was set to 30 and iterations were set to 80 in this study.

It can be seen from formula (10) that there is exponential relationship between state transition probability, pheromone trails, and heuristic information. Thus, the probability is more sensitive to these factors with larger parameters α, β, and γ. To verify the influence of these parameters, preliminary tests were done to choose the parameters levels as well. $\alpha = 1$ was used as a reference level to study the effect of parameters β and γ on makespan in the interval of $[1, 10]$. The results are shown in Figure 4.

It can be observed from Figure 4 that medium β and smaller γ should be used for instances of $S2$ categories with small jobs to obtain better makespan while the larger γ should be adopted for instances of $S3$ categories with large jobs on the contrary. For instances of $S1$ categories with mixed job sizes, smaller values for β and medium γ seem better compared with the former two categories. All instances generate bad results when β is too close to 1. Large size jobs usually have lower flexibility when arranged in batches compared with small size jobs. So high level of γ is used to arrange large size jobs first with high probability for instances with large jobs. Similarly, big β is used to batch jobs with similar processing time together to increase the efficiency of batch processing time. According to the analysis above, three levels for each parameter are selected in this study as shown in Table 2.

Pheromone trails evaporate at each iteration and the speed is controlled by the parameter ρ. $(1 - \rho)$ means evaporation rate. A high evaporation rate leads to a constant change of pheromone trails on each solution element while a low rate results in the pheromone trails on solution elements that cannot evaporate in time.

Pheromone trails of each solution element is limited in the interval of $[\tau_{min}, \tau_{max}]$ in MMAS algorithm. The pheromone trails are usually set to a high value of τ_{max} at the beginning of running in order to improve the searching ability in solution space. If a solution element is always in the state of evaporation and its pheromone trail decreases to τ_{min} just after Nc iterations then we have $\tau_{min} = \tau_{max} \cdot \rho^{Nc}$. Since $\tau_{min} = \tau_{max}/2n$ as mentioned above, it can be derived that $\rho^{Nc} = 1/2n$; that is, $Nc = \log_{\rho}^{1/2n}$, $0 < \rho < 1$, where n is the number of jobs. The relationship between Nc and ρ for 100 jobs is described in Figure 5.

It can be seen from Figure 5 that Nc increases dramatically when $\rho > 0.6$; thus ρ is set to 0.6 in this study in order to ensure the pheromone trails evaporate not too quickly or

Table 3: Results of MMAS compared with CPLEX.

Problem category	MMAS					CPLEX
	Min	Max	Avg	Optimal (%)	AvgTime (s)	AvgTime (s)
*S*1						
*J*1*P*1	20.00	60.00	36.64	100.00	0.11	0.64
*J*1*P*2	33.00	127.00	71.98	100.00	0.10	0.63
*J*2*P*1	43.00	105.00	69.13	100.00	0.33	1.18
*J*2*P*2	81.00	198.00	131.70	96.00	0.33	1.42
*S*2						
*J*1*P*1	13.00	29.00	20.90	100.00	0.22	0.44
*J*1*P*2	23.00	58.00	40.24	100.00	0.22	0.45
*J*2*P*1	28.00	48.00	37.97	100.00	0.81	28.04
*J*2*P*2	46.00	94.00	71.94	99.00	0.82	26.90
*S*3						
*J*1*P*1	27.00	64.00	44.12	100.00	0.08	0.59
*J*1*P*2	50.00	124.00	85.53	100.00	0.08	0.58
*J*2*P*1	56.00	122.00	87.84	100.00	0.23	1.07
*J*2*P*2	106.00	217.00	167.18	100.00	0.23	1.12

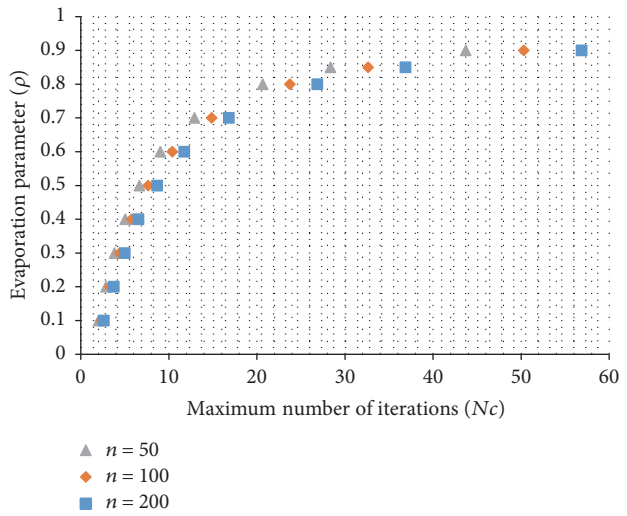

Figure 5: Relationship between Nc and ρ.

slowly and the solution space can be explored in a reasonable run time.

4.3. Performance Comparison for Small Scale Instances. As shown in Table 3, the results obtained from MMAS is compared with that from CPLEX (a commercial solver for linear and mixed-integer problems). Due to the NP-hardness of the problem, only small scale instances of $J1$ and $J2$ problem categories are solved by using CPLEX. The minimum, maximum, and average value of $C_{\max}$ reported from MMAS are listed in columns 3 to 5. The average run time of MMAS and CPLEXT are given in columns 7 and 8. Column 6 indicates the percent of the optimal values obtained by using MMAS compared with the results from CPLEX.

Nearly all small scale instances can be solved optimally by using MMAS except several instances from $J2P2S1$ and $J2P2S2$ with relative larger solution space. MMAS is competitive in computational time compared with CPLEX for all instances. Computational time of all instances from MMAS is less than 1 second. The computational time of CPLEX varies greatly depending on the solution space of an instance.

4.4. Algorithm Performance Evaluation. To evaluate the performance of the algorithm for all problem categories, MMAS designed in this paper was compared with GA [9]. A basic ant algorithm ant cycle (AC) was coded to solve the problem as well and the parameters used were as prescribed in Cheng et al. [21]. Two well-known heuristics FFLPT and BFLPT were also compared with MMAS in the experiment study. 100 instances were generated for each problem category and the best result of 10 runs for each instance was eventually used.

Comparison of various algorithms is summarized in Table 4. Column 1 in Table 4 presents the run code. Column 2 shows the results of MMAS compared with other algorithms where B, E, and I mean the makespan obtained by MMAS are better than, equal to, or inferior to that of other algorithms, respectively. Columns 3–6 report the proportion that the makespan obtained by using MMAS compared with that of other algorithms in 100 instances of $S1$ problem categories. Columns 7–10 and columns 11–14 are similar to columns 3–6. For example, the first number of 0.01 indicates that there is a proportion of 0.01 results reported by MMAS better than that generated by AC. As the problem under study is strongly NP-hard, the solution space is to explode along with the population size increasing form $J1$ to $J4$. On the other hand, smaller job sizes give more combination of batches; thus the solution space becomes smaller for $S3$ problem categories compared with $S1$ for a given number of jobs.

It can be seen from Table 4 that MMAS outperforms other algorithms on the whole. Several results reported by MMAS inferior to those generated by other algorithms have

Table 4: Results of MMAS compared with other algorithms.

Run code	Cp.	$S1$				$S2$				$S3$			
		AC	GA	BFLPT	FFLPT	AC	GA	BFLPT	FFLPT	AC	GA	BFLPT	FFLPT
$J1P1$	B	0.01	0.02	0.16	0.17	0.00	0.10	0.16	0.16	0.00	0.01	0.07	0.07
	E	0.99	0.98	0.84	0.83	1.00	0.90	0.84	0.84	1.00	0.99	0.93	0.93
	I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J1P2$	B	0.02	0.02	0.10	0.13	0.01	0.14	0.19	0.19	0.00	0.03	0.08	0.11
	E	0.98	0.98	0.90	0.87	0.99	0.86	0.81	0.81	1.00	0.97	0.92	0.89
	I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J2P1$	B	0.04	0.45	0.47	0.59	0.01	0.31	0.32	0.32	0.00	0.07	0.14	0.20
	E	0.96	0.55	0.53	0.41	0.99	0.69	0.68	0.68	1.00	0.93	0.86	0.80
	I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J2P2$	B	0.13	0.42	0.41	0.57	0.06	0.32	0.33	0.34	0.00	0.14	0.19	0.29
	E	0.87	0.58	0.59	0.43	0.94	0.68	0.67	0.66	1.00	0.86	0.81	0.71
	I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J3P1$	B	0.43	0.84	0.83	0.93	0.23	0.89	0.91	0.91	0.00	0.30	0.30	0.43
	E	0.57	0.16	0.17	0.07	0.75	0.11	0.09	0.09	1.00	0.70	0.70	0.57
	I	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J3P2$	B	0.60	0.90	0.85	0.99	0.39	0.87	0.87	0.87	0.12	0.27	0.27	0.39
	E	0.40	0.10	0.15	0.01	0.60	0.13	0.13	0.13	0.88	0.73	0.73	0.61
	I	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J4P1$	B	0.78	0.93	0.90	1.00	0.49	0.99	0.99	0.99	0.05	0.18	0.18	0.30
	E	0.21	0.06	0.09	0.00	0.49	0.01	0.01	0.01	0.95	0.82	0.82	0.70
	I	0.01	0.01	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$J4P2$	B	0.92	0.95	0.92	0.98	0.51	0.99	0.99	0.99	0.35	0.22	0.22	0.30
	E	0.05	0.04	0.06	0.02	0.43	0.00	0.00	0.00	0.65	0.78	0.78	0.70
	I	0.03	0.01	0.02	0.00	0.06	0.01	0.01	0.01	0.00	0.00	0.00	0.00

been earmarked by rectangle. For $J1$ problem categories, results obtained from MMAS are mostly equal to that from other algorithms. A percentage of about 16% results from MMAS is better than that from algorithms BFLPT and FFLPT for $S1$ and $S2$ problem categories. The percentage is 7% for $S3$ problem categories, which is because more results are the same for all algorithms in a small solution space. Up to 14% reported by MMAS are better than GA for $J1P2S3$ problem category. The advantage of MMAS to algorithms like GA, BFLPT, and FFLPT is much more remarkable for $J2$ problem categories. More better results can be found by MMAS than AC and the percentage is up to 6% to 13% for $J2P2$ problem categories. More than 80% results obtained from MMAS are better than that from GA, BFLPT, and FFLPT for $J3S1$ and $J3S2$ problem categories. About 23% to 43% percent of results from MMAS are better than AC while the latter reports few better results. For $J4$ problem categories, more than 90% results generated by MMAS are better than that from algorithms AC, GA, BFLPT, and FFLPT for large job population instances. Almost all other algorithms report several better results than MMAS for $J4$ problem categories.

Based on the analysis given above, MMAS possesses higher ability in solution space exploring. It outperforms all other algorithms for almost all instances especially for large job population. As MMAS is exploring solution space in probability, the optimal solution cannot be guaranteed with a limited population and iterations. Therefore, several results obtained from MMAS are inferior to other algorithms even to effective heuristics like BFLPT and FFLPT.

The makespan increases for $P2$ problem categories compared with $P1$ problem categories but there is no significant performance change for various algorithms.

To illustrate the gap between MMAS and other algorithms for different population size, the average percentage of results with better makespan obtained by MMAS compared with other algorithms is given in Figure 6.

Averagely, a percentage of 10% results obtained from MMAS are better than that from other algorithms for $J1$ problem categories while the number is 70% for $J4$. Algorithm AC is the second best algorithm compared to GA and the other two heuristics. BFLPT and FFLPT are all effective in solving the problem especially for small population size. BFLPT is generally better than FFLPT for almost all problem categories and BFLPT is also used in GA to generate initial solutions.

5. Conclusions and Future Work

The problem of scheduling single batch processing machine was studied in this paper. An improved ant algorithm MMAS (Max–Min Ant System) was designed and applied to solve the problem. Parameters of MMAS were determined by preliminary experiment and a local search method MJE was also presented to improve the performance of MMAS.

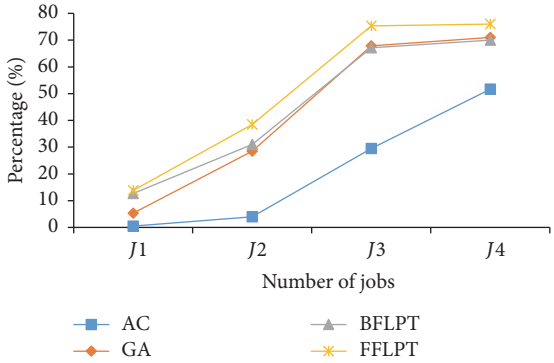

Figure 6: Average percentage of results obtained from MMAS compared with that from other algorithms.

Optimal objectives can be obtained by using MMAS in less computational time compared to CPLEX for small scale problems. To evaluate the performance of MMAS, it was compared with basic ant algorithm AC (Ant Cycle), GA (Genetic Algorithm), and the other two well-known heuristics of scheduling batch processing machine, that is, FFLPT (First Fit Longest Processing Time) and BFLPT (Best Fit Longest Processing Time). The numeric experiment showed that MMAS outperformed other algorithms for almost all instances in various problem categories especially for larger population size.

Considering the good performance of MMAS, better heuristics mechanism can be designed and applied to other problems of scheduling batch processing machine. The problem under study with other objectives and problem constraints including release times and due dates can be studied as well.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (no. 2014QNA48) and National Natural Science Foundation of China (no. 71401164).

References

[1] Y. Ikura and M. Gimple, "Efficient scheduling algorithms for a single batch processing machine," *Operations Research Letters*, vol. 5, no. 2, pp. 61–65, 1986.

[2] 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.

[3] C. S. Sung and Y. I. Choung, "Minimizing makespan on a single burn-in oven in semiconductor manufacturing," *European Journal of Operational Research*, vol. 120, no. 3, pp. 559–574, 2000.

[4] R. Uzsoy, "Scheduling a single batch processing machine with non-identical job sizes," *International Journal of Production Research*, vol. 32, no. 7, pp. 1615–1635, 1994.

[5] L. Dupont and F. Jolai Ghazvini, "Minimizing makespan on a single batch processing machine with non-identical job sizes," *Journal Européen des Systèmes Automatisés*, vol. 32, no. 4, pp. 431–440, 1998.

[6] L. Dupont and C. Dhaenens-Flipo, "Minimizing the makespan on a batch machine with non-identical job sizes: An exact procedure," *Computers & Operations Research*, vol. 29, no. 7, pp. 807–819, 2002.

[7] X. Li, Y. Li, and Y. Wang, "Minimising makespan on a batch processing machine using heuristics improved by an enumeration scheme," *International Journal of Production Research*, vol. 55, no. 1, pp. 176–186, 2017.

[8] S. Melouk, P. Damodaran, and P.-Y. Chang, "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing," *International Journal of Production Economics*, vol. 87, no. 2, pp. 141–147, 2004.

[9] P. Damodaran, P. Kumar Manjeshwar, and K. Srihari, "Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms," *International Journal of Production Economics*, vol. 103, no. 2, pp. 882–891, 2006.

[10] A. Husseinzadeh Kashan, M. Husseinzadeh Kashan, and S. Karimiyan, "A particle swarm optimizer for grouping problems," *Information Sciences*, vol. 252, pp. 81–95, 2013.

[11] P. Damodaran, O. Ghrayeb, and M. C. Guttikonda, "GRASP to minimize makespan for a capacitated batch-processing machine," *The International Journal of Advanced Manufacturing Technology*, vol. 68, no. 1-4, pp. 407–414, 2013.

[12] S. Zhou, M. Liu, H. Chen, and X. Li, "An effective discrete differential evolution algorithm for scheduling uniform parallel batch processing machines with non-identical capacities and arbitrary job sizes," *International Journal of Production Economics*, vol. 179, pp. 1–11, 2016.

[13] L. Jiang, J. Pei, X. Liu, P. M. Pardalos, Y. Yang, and X. Qian, "Uniform parallel batch machines scheduling considering transportation using a hybrid DPSO-GA algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 89, no. 5-8, pp. 1887–1900, 2017.

[14] T. Stützle and H. H. Hoos, "Max-min ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[15] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[16] M. Dorigo and C. Blum, "Ant colony optimization theory: a survey," *Theoretical Computer Science*, vol. 344, no. 2-3, pp. 243–278, 2005.

[17] R. Xu, H. Chen, and X. Li, "Makespan minimization on single batch-processing machine via ant colony optimization," *Computers & Operations Research*, vol. 39, no. 3, pp. 582–593, 2012.

[18] L. D'Acierno, M. Gallo, and B. Montella, "An ant colony optimisation algorithm for solving the asymmetric traffic assignment problem," *European Journal of Operational Research*, vol. 217, no. 2, pp. 459–469, 2012.

[19] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 3, pp. 299–325, 1974.

[20] M. Pirlot, "General local search methods," *European Journal of Operational Research*, vol. 92, no. 3, pp. 493–511, 1996.

[21] B.-Y. Cheng, H.-P. Chen, and S.-S. Wang, "Improved ant colony optimization method for single batch-processing machine with non-identical job sizes," *Journal of System Simulation*, vol. 21, no. 9, pp. 2687–2695, 2009.