sustainability

MDPI

Article

An Enhanced MOPSO Algorithm for Energy-Efficient Single-Machine Production Scheduling

Yueyue Liu, Xiaoya Liao and Rui Zhang *

School of Economics & Management, Xiamen University of Technology, Xiamen 361024, China; yhly429@126.com (Y.L.); lxyjiang@gmail.com (X.L.)

* Correspondence: r.zhang@ymail.com

Received: 2 August 2019; Accepted: 23 September 2019; Published: 28 September 2019

Abstract: In recent years, the concerns on energy efficiency in manufacturing systems have been growing rapidly due to the pursuit of sustainable development. Production scheduling plays a vital role in saving energy and promoting profitability for the manufacturing industry. In this paper, we are concerned with a just-in-time (JIT) single machine scheduling problem which considers the deterioration effect and the energy consumption of job processing operations. The aim is to determine an optimal sequence for processing jobs under the objective of minimizing the total earliness/tardiness cost and the total energy consumption. Since the problem is $\mathcal{NP}$-hard, an improved multi-objective particle swarm optimization algorithm enhanced by a local search strategy (MOPSO-LS) is proposed. We draw on the idea of k-opt neighborhoods and modify the neighborhood operations adaptively for the production scheduling problem. We consider two types of k-opt operations and implement the one without overlap in our local search. Three different values of k have been tested. We compare the performance of MOPSO-LS and MOPSO (excluding the local search function completely). Besides, we also compare MOPSO-LS with the well-known multi-objective optimization algorithm NSGA-II. The experimental results have verified the effectiveness of the proposed algorithm. The work of this paper will shed some light on the fast-growing research related to sustainable production scheduling.

Keywords: energy efficiency; production scheduling; just-in-time; multi-objective particle swarm optimization; local search

1. Introduction

The manufacturing industry has been regarded as an intensive energy consumer. To achieve sustainable development goals, a number of regulations are urging the manufacturers to adopt energy-saving measures. In addition to an upgrade of production equipment, which is usually costly, the possibility of using soft techniques to achieve energy savings should be emphasized. In fact, production scheduling could play a significant role in reducing the energy consumption of manufacturing processes. However, the traditional production scheduling software, which is often embedded in the company's ERP or MES, has been designed solely for the productivity or profitability goal, without consideration of sustainability impacts. To take energy consumption and other environmental factors into consideration, the scheduling models and algorithms need to be significantly revised or even rewritten. In the redevelopment process, a major difficulty is that the energy model, which characterizes the power consumption of manufacturing activities, is highly industry-dependent and factory-dependent, and therefore, energy-efficient production scheduling algorithms are not universally applicable and have to be designed specifically for each company. Currently, some energy-intensive manufacturers (like steel companies) have adopted energy-saving production scheduling techniques, and the procedure is waiting to be extended and applied to other

industries. Another issue that prevents the application of such scheduling technologies is the high computational complexity. The simultaneous consideration of energy-related aspects will complicate the scheduling model (leading to more constraints and variables), which requires more powerful optimization algorithms to solve. This poses a theoretical challenge to the operations research and management science community.

Energy-efficient production scheduling problems have been extensively studied by a number of researchers in recent years [1–4]. In the existing research, considerations for energy efficiency can be divided into two categories, i.e., energy-related objectives and energy-related constraints. Wang et al. [1] study a bi-objective single machine batch scheduling problem with non-identical job sizes. Zhang and Chiong [5] propose an enhanced multi-objective genetic algorithm to solve the job shop scheduling problem minimizing total energy consumption and total weighted tardiness. Wu et al. [6] investigate the flexible job-shop scheduling problem with the objective of minimizing total energy consumption and makespan. They consider the deterioration effect as well. Liao et al. [7] use MOPSO to address the bi-objective single machine scheduling problem with energy consumption constraints. Módos et al. [8] consider a production scheduling problem with large electricity consumption. The energy-related constraint requires that, in the specified time intervals, the total energy consumption should not exceed a given limit.

Just-in-time (JIT) objectives are commonly considered in production scheduling problems. Under the JIT logic, both earliness and tardiness (with respect to due dates) should be penalized [9,10]. The former can cause inventory holding costs such as storage cost and extra delivering cost (especially for perishable goods), while the latter can cause loss of reputation, sale and may even cause penalty.

In real-world manufacturing systems, the processing times of certain jobs can be longer than their nominal values because of the deterioration effect (deterioration effect refers to the phenomenon that a job requires longer processing time if its starting time is postponed or later than expected). There are varieties of factors that may cause deterioration such as high workload of machines, insertion of maintenance activities, and fatigue of human operators [11,12]. Deterioration can also be explained by industry-specific reasons, for example, a steel slab in the hot rolling production process has to be reheated (which takes extra processing time) if it has been kept waiting for a long time which brings a significant decline in temperature [13].

To the best of our knowledge, there are no results in the existing literature for production scheduling problems that integrate the JIT objective, the energy-saving requirement and explicit consideration of the deterioration effect. Production systems with the above characteristics are commonly observed in mechanical and metal manufacturing companies. Note that the single-machine JIT scheduling problem with a deterioration effect is $\mathcal{NP}$-hard [14], which means the time required by an exact algorithm to solve the problem increases exponentially with the number of jobs. Hence, a multi-objective particle swarm optimization algorithm enhanced by local search (MOPSO-LS) is presented in this paper to obtain near-optimal solutions for the discussed problem.

The rest of this paper is organized as follows. Section 2 introduces the basic definitions and notations for the bi-objective single-machine scheduling problem with deterioration. Section 3 describes the proposed MOPSO-LS algorithm. Computational results are shown in Section 4. Conclusions and future work are presented in Section 5.

2. The Scheduling Problem

2.1. Problem Statement

The problem is described as follows. A set of jobs labeled as $J = \{j | j = 1, 2, \ldots, n\}$ are to be scheduled on a single machine. The aim is to obtain the best sequence for processing the set of jobs to minimize the total weighted earliness/tardiness ($TWET$) and the total energy consumption (TEC). Each job j is characterized by a basic processing time tbp_j, a latest starting time tls_j, a weight w_j and a due date d_j. For a given sequence of jobs, the completion time tc_j of job j can be determined,

and the earliness/tardiness of job j is defined as $et_j = \left|tc_j - d_j\right|$. One objective is to minimize the total weighed earliness/tardiness $TWET = \sum_{j=1}^{n} w_j et_j$. Another objective is to minimize the total energy consumption $TEC = \sum_{j=1}^{n} tp_j e_j$, where e_j is the power consumption for processing job j per unit time and tp_j is the actual processing time of job j.

With the deterioration effect in consideration, the actual processing time of each job j is variable and determined by the basic processing time tbp_j and the starting time ts_j. For each job j, when the starting time ts_j is earlier than or equal to its latest starting time tls_j, the actual processing time tp_j is equal to its basic processing time tbp_j. Otherwise, the increment in actual processing time tp_j is proportional to the delay of starting time, i.e., $ts_j - tls_j$. Therefore, tp_j can be calculated as follows:

$$tp_j = tbp_j + b_j \max\{0, ts_j - tls_j\}, \tag{1}$$

where b_j is the deterioration coefficient of job j.

The other assumptions for the problem are stated as follows (these are standard assumptions for the modeling of scheduling problems and are normally satisfied in practical situations):

- All jobs are available at time zero.
- The machine can handle only one job at a time.
- There are no precedence relations between the jobs.
- Preemption is not allowed.
- Setup times are not considered.

2.2. A Small Example

Consider a set of eight jobs labeled as $j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. The information about each job is shown in Table 1. Obviously, each possible sequence of the jobs corresponds to a feasible solution, so the number of feasible solutions for this instance is: $8! = 40,320$. Figure 1 shows the objective space of this instance and marks all the non-dominated solutions. In this case, there are eight non-dominated solutions in total, which constitute the Pareto front, i.e., the set of Pareto-optimal solutions (we say a solution is Pareto-optimal if neither objective can be improved without sacrificing the other objective).

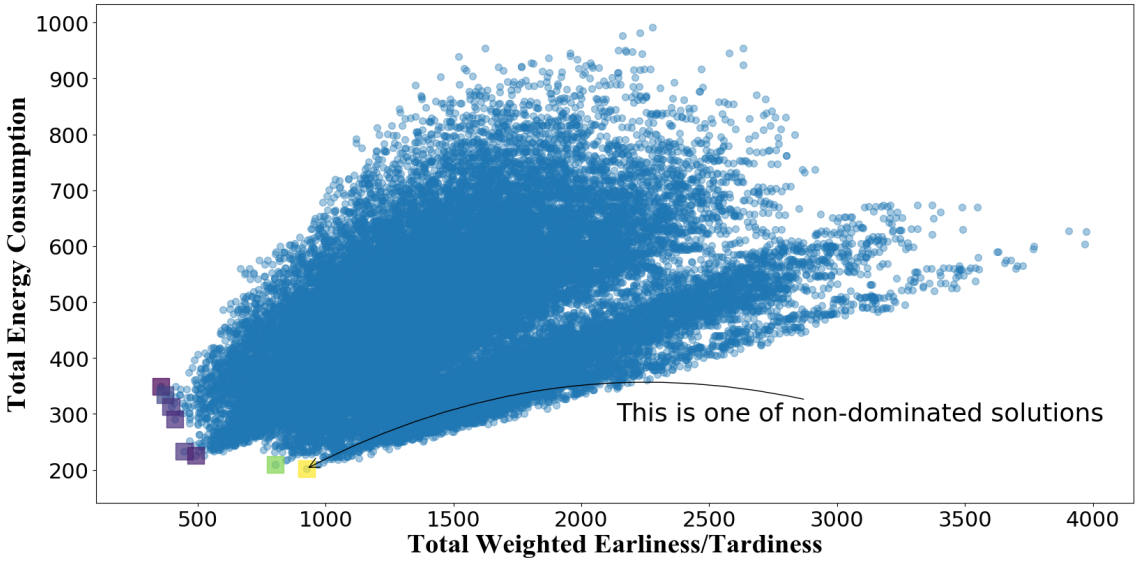


Figure 1. Feasible solutions of the instance.

Table 1. The job data of a small-scale instance.

j	tbp_j	tls_j	e_j	b_j	d_j	w_j
1	5	12	3	0.9818	5	3
2	3	16	2	0.4282	30	2
3	10	11	3	0.3046	29	5
4	4	6	1	0.3868	12	9
5	6	5	5	0.2814	19	6
6	6	8	5	0.6004	15	5
7	6	21	3	0.8456	25	1
8	7	21	2	0.8839	34	6

3. The Proposed Algorithm

3.1. The Basic PSO Algorithm

The proposed algorithm adopts MOPSO as the main framework and incorporates an enhanced local search method to promote efficiency. We first make a brief introduction to the basic principles of PSO. The PSO algorithm is a population-based stochastic optimization technique inspired by the social behavior of bird flocking or fish schooling. It is a well-known meta-heuristic algorithm for solving both continuous and combinatorial optimization problems.

Firstly, the algorithm generates a population of particles and the position of each particle represents a potential solution in the search space for the considered problem. According to the evaluation of the objective function to be optimized, each particle has a fitness value. Then, the fitness determines the flying direction and speed of the particles. In each iteration, two types of best positions are preserved. One is called the personal best position (p_{best}) which represents the best solution that each particle has achieved so far. The other is called the global best position (g_{best}) which represents the best position obtained so far by all the particles. Finally, the particles learn from these two best positions based on their current flying routes, and the positions of the particles get updated in each iteration. The iterations continue until the convergence criterion is satisfied.

3.2. Encoding and Decoding

We use the largest position value (LPV) rule as the decoding method, which transforms continuously encoded particles into discrete solutions. Simply speaking, the decoded solution records the relative order of the corresponding position values (from largest to smallest). For example, for a particle $pt = (2.38, -0.55, -3.64, 3.98, 1.68, -1.26, 0.8, -2.33)$, its corresponding job sequence is $S_{pt} = (2, 5, 8, 1, 3, 6, 4, 7)$, because the fourth value in pt is the largest (rank 1) and the third value in pt is the smallest (rank 8).

3.3. Solution Initialization

Generally, the initial population is generated by specific heuristic rules combined with some random methods to get high-quality and distinct solutions. However, to encourage solution diversity and to facilitate the evaluation of the k-opt neighborhood operations, the individuals of the initial population are completely randomized in our proposed algorithm.

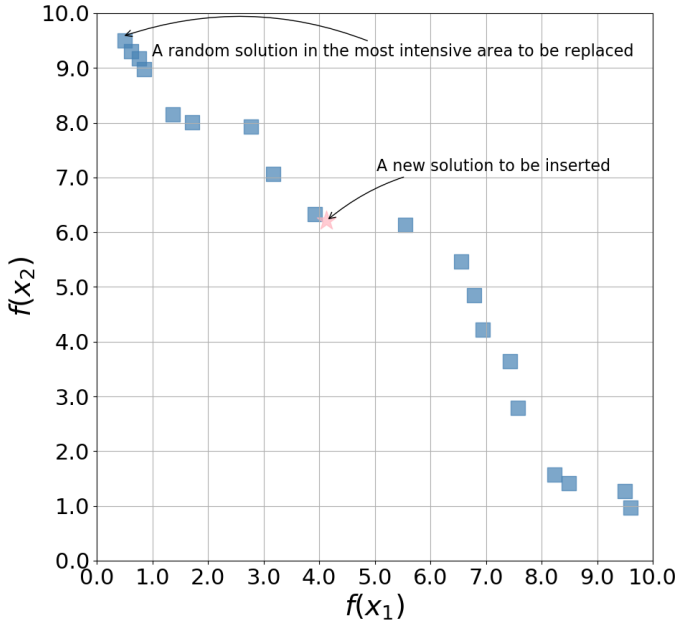
3.4. Sorting of Solutions

In multi-objective optimization settings, Pareto dominance is the main criterion for distinguishing the quality of different solutions. For our research, both objectives aim at finding the minimum. In this case, solution $\boldsymbol{x}_1$ with objective values $(TWET_1, TEC_1)$ is said to dominate solution $\boldsymbol{x}_2$ with objective values $(TWET_2, TEC_2)$ if either one of the following two conditions is satisfied:

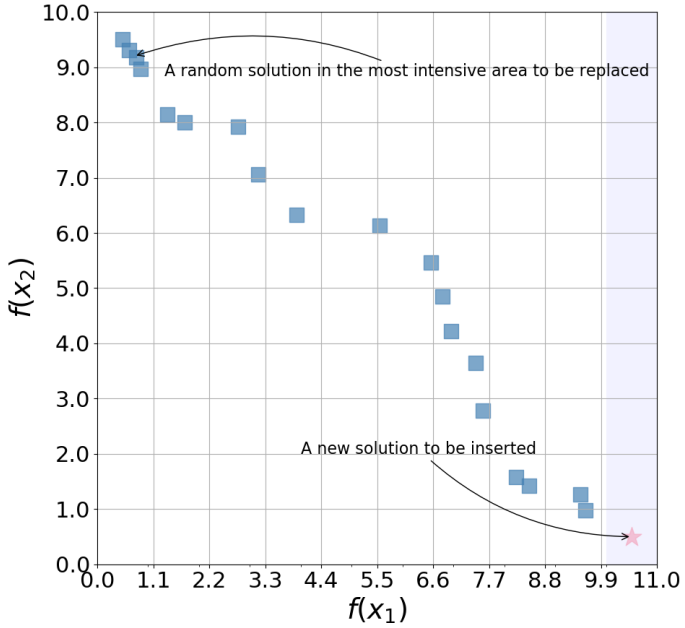
- $TWET_1 < TWET_2$ and $TEC_1 \leq TEC_2$
- $TWET_1 \leq TWET_2$ and $TEC_1 < TEC_2$

The relation will be denoted by $x_1 \prec x_2$. Two solutions are treated equally in the same Pareto rank if they are mutually non-dominated. All the non-dominated solutions are stored in an external archive.

In addition to Pareto dominance, the adaptive grid method proposed by Coello Coello [15] is used to sort solutions that are not mutually dominated. The aim is to avoid over-crowded solutions and obtain evenly distributed non-dominated solutions for the decision maker. Figure 2 shows the insertion of a new solution in the adaptive grid with grid size $n_d = 10$.



(**a**)



(**b**)

Figure 2. Example of the insertion of a new solution. (**a**) The case where the inserted individual lies within the boundaries of the grid; (**b**) The case where the inserted individual lies outside the previous boundaries of the grid.

3.5. External Repository and Updating Mechanisms

In standard PSO intended for single-objective optimization, either the personal best solution associated with each particle or the global best solution is an individual solution. However, it is necessary to preserve a number of equally optimal (non-dominated) solutions when addressing multi-objective optimization problems. For this purpose, we build an external repository (denoted by a set ϕ) to save all the historical non-dominated solutions found in the search process.

The maintenance rules are described as follows. In each iteration, if the newly produced solutions are all dominated by some existing solutions in ϕ, nothing happens. If there are any new non-dominated solutions, then insert all of them into the repository and the dominated solutions originally in the repository are eliminated.

3.6. The Mutation Operator

We apply a mutation operator to enhance the global searching capability of MOPSO (the concept of mutation is adapted from genetic algorithm, representing a perturbation technique which alters the current solutions slightly in the hope of finding better solutions). The mutation operator is applied to the particles of the swarm after the position updating process. Specifically, we use a mutation index (represented as $mutIndex$) to control the probability of mutation. When $mutIndex$ is a positive real number, the mutation operator will be applied to each particle with a probability of $mutIndex$. In the selected particle, the mutation operator changes the value of a randomly identified position to a real number generated from the uniform distribution whose interval is centered at its original value with $mutRadius$ as its radius. The $mutIndex$ and $mutRadius$ mentioned above can be calculated as follows:

$$mutIndex = \left(1 - \frac{iter}{iterMax \times p_m}\right)^{\frac{3}{2}} \tag{2}$$

$$mutRadius = \frac{rangeMax \times mutIndex}{2}, \tag{3}$$

where $iter$ and $iterMax$ represents the current iteration number and the maximum iteration number of MOPSO, p_m is the mutation rate defined in MOPSO, and $rangeMax$ is set at 4 in our algorithm.

3.7. Local Search

The two-opt neighborhood operator, which reverses one segment of the sequence, is the most commonly used operator to generate neighboring solutions for routing problems. The idea can be extended to $k-1$ segments (k-opt). However, the operator needs to be modified in our research since a scheduling problem is different from a routing problem.

We define the k-opt neighborhood operations as follows. For a given sequence, we select $k-1$ segments randomly, and then we reverse some of these segments. Since each of the segments may or may not be reversed, there are a number of possible combinations. If the number of selected segments is $k-1$, there are $\sum_{i=1}^{k-1} C_{k-1}^{i}$ ways to rearrange a schedule. For example, when considering 2 segments (3-opt), there are three possible outcome solutions (Figure 3). Similarly, for the case of three segments (four-opt), there are seven possible outcome solutions.

It is supposed that the selected segments do not have any overlap, because in the case of overlap, the order of reversing the segments will affect the final solutions. For example, consider two segments with an overlap (shown in Figure 4). For the original sequence $S_0 = (1,2,3,4,5,6,7,8)$, the first segment is from the second job to the sixth job and the second segment is from the fifth job to the seventh job. If we reverse the first segment first, we get the sequence $S_1 = (1,6,5,4,3,2,7,8)$, and then we reverse the second segment which is currently filled with jobs $(3,2,7)$. However, if we reverse the second segment first, we get the sequence $S_2 = (1,2,3,4,7,6,5,8)$, and then we reverse the first segment which is currently filled with jobs $(2,3,4,7,6)$. The resulting solutions will certainly be

different in these two cases. To avoid confusion, we assume the selected segments for *k*-opt operations are overlap-free.

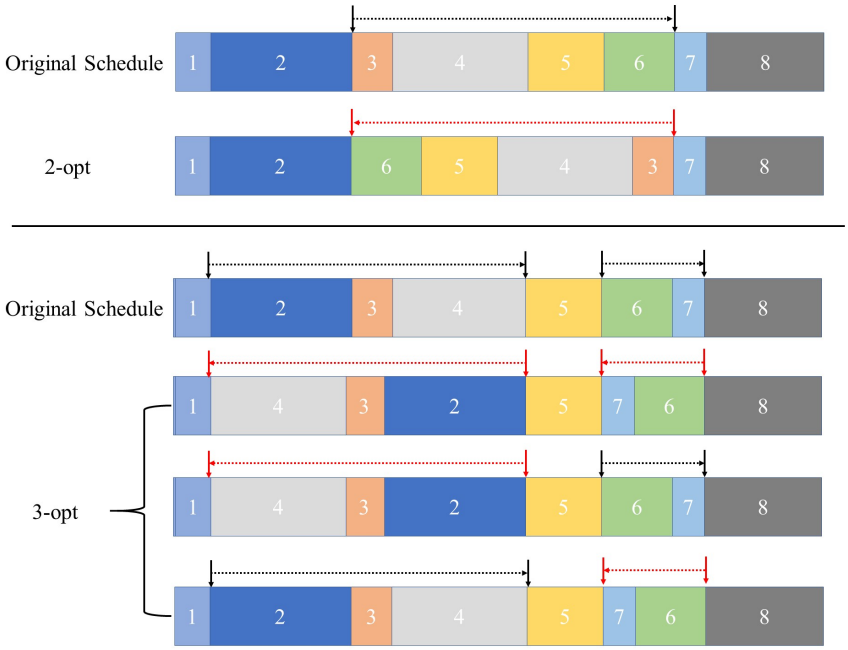


Figure 3. All possible cases for two-opt and three-opt without overlap.

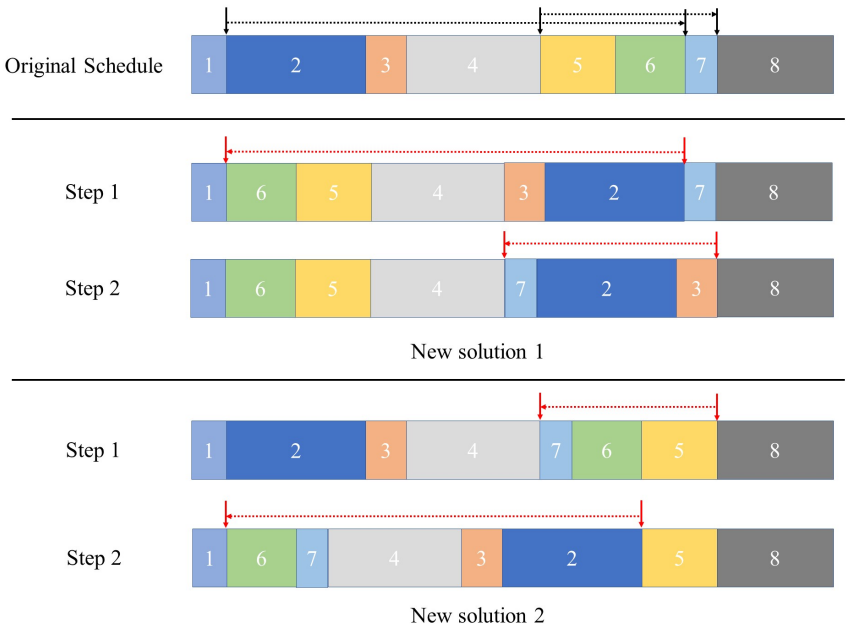


Figure 4. Two possible cases for two-opt with overlap.

After the solution set ϕ in external repository has been updated in each iteration, the *k*-opt operator is performed on one of the solutions in ϕ. All solutions have equal probabilities of being selected. The new solutions produced by this operation will be added to ϕ. Then all the solutions will be evaluated and saved according to the updating mechanisms introduced before.

3.8. MOPSO-LS Framework

The entire algorithm structure of the MOPSO-LS is shown in Figure 5. Compared with the standard PSO workflow, our algorithm features a novel mutation operator and local search module, which greatly enhances the exploitation ability for coping with large-scale search space. In addition, the mechanisms for handling multi-objective optimization are also important contributors to the overall algorithm but not directly reflected in the flowchart.

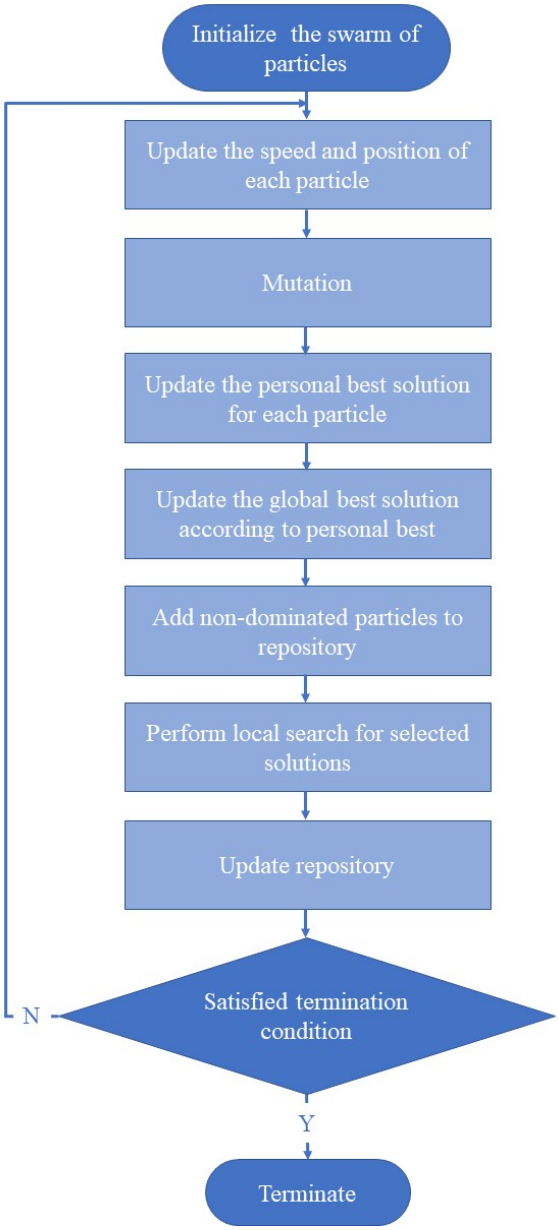


Figure 5. Main steps of the proposed multi-objective particle swarm optimization algorithm enhanced by local search (MOPSO-LS).

4. Computational Experiments

4.1. Experimental Settings

Because there is no standard data set for the studied problem, we followed the data generation rules in literature for scheduling problems with deteriorating jobs and scheduling problems with

energy consumption considerations. The parameters generated for the studied scheduling problem are described as follows and summarized in Table 2.

- The basic processing time tbp_j of each job is generated uniformly from $unif(1, 100)$.
- The deterioration factor of each job b_j is generated from the uniform distribution $unif(0, 1)$.
- The latest starting time of each job $tls_j = 30 \times n \times unif(0, 1)$ where n is the number of jobs.

Table 2. Data generation rules.

Item	Distribution
tbp_j	$tbp_j \sim unif(1, 100)$
b_j	$b_j \sim unif(0, 1)$
tls_j	$tls_j \sim 30 \times n \times unif(0, 1)$
d_j	$d_j \sim 80 \times n \times unif(0, 1)$
n	$n \in \{15, 30, 50, 75, 100\}$

The true Pareto front of each instance was not known, but for comparison purpose, we generated an approximate Pareto front for each instance by the following steps. First, all algorithms including MOPSO-LS (2-opt), MOPSO-LS (3-opt), MOPSO-LS (4-opt), MOPSO (excluding local search) and NSGA-II were repeated 20 times. Then, we identified all the non-dominated solutions from the solutions found by the above algorithms and these non-dominated solutions constitute the approximate Pareto front.

4.2. The Performance Measures

Convergence to the Pareto front and evenness of solution distribution were two main criteria for assessing multi-objective optimization algorithms. In accordance with these requirements, we adopt the generational distance (GD) measure [16] and the spacing (SP) measure [17] to evaluate the performance of algorithms.

GD measures the distance between the set of non-dominated solutions found by the algorithm and the (approximate) Pareto front [16]. GD is defined as follows:

$$GD = \frac{\sqrt{\sum_{i=1}^{m} d_i^2}}{m}, \tag{4}$$

where m is the number of non-dominated solutions found and d_i is the Euclidean distance between solution i and its nearest counterpart solution from the Pareto front.

SP measures the spread (distribution evenness) of the solutions found [17]. SP is defined as follows:

$$SP = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (\bar{d} - \tilde{d}_i)^2}, \tag{5}$$

where $\bar{d} = \sum_{i=1}^{m} \tilde{d}_i / m$ and $\tilde{d}_i$ is the Manhattan distance between solution i and its nearest neighbor solution in the set of non-dominated solutions found, formally defined as follows:

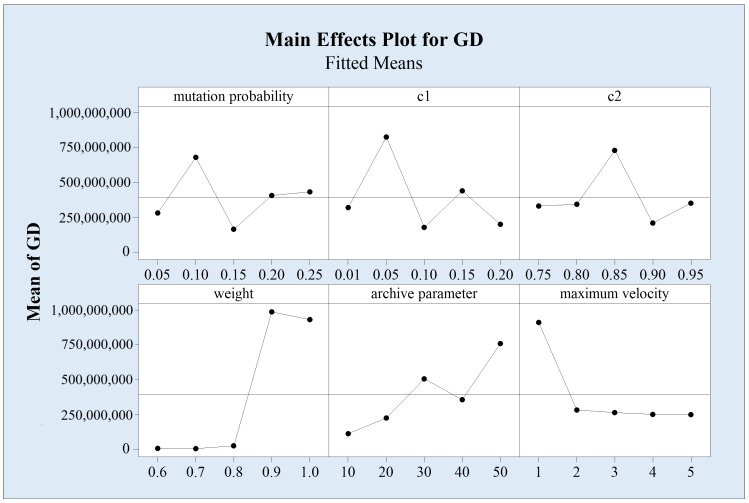
$$\tilde{d}_i = \min_{j, j \neq i} \{|f_1(\boldsymbol{x}_i) - f_1(\boldsymbol{x}_j)| + |f_2(\boldsymbol{x}_i) - f_2(\boldsymbol{x}_j)|\}. \tag{6}$$

4.3. Parameter Tuning Experiments

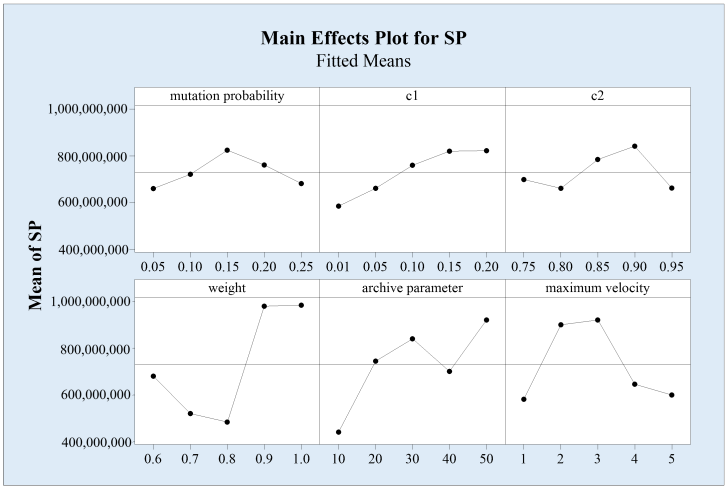
The setting of parameters can have a vital impact on the performance of algorithms. In the proposed MOPSO-LS, the weight ω, learning factors c_1, c_2, mutation probability p_m, maximum velocity $V_{\max}$ and the grid size n_d [15] are important parameters. To find the best settings, we applied a design of experiments (DOE) approach to study the influence of these six parameters.

The adopted DOE approach was based on the Taguchi method [18]. All the parameter settings for MOPSO and MOPSO-LS were the same. The details of parameters are shown in Table 3. We also performed the DOE for the NSGA-II to ensure a fair comparison. For the NSGA-II, we focused on the crossover probability p_c, mutation probability p_{m2}, crossover distribution index di_c and mutation distribution index di_m. The details are shown in Table 4.

A randomly generated instance with 50 jobs was used in the experiments. For each combination of parameter values, we ran the corresponding algorithm for 10 times and record the average values of GD and SP. The results are shown in Figures 6 and 7. In conclusion, the optimal settings of parameters for MOPSO and MOPSO-LS are as follows: $\omega = 0.7$, $c_1 = 0.1$, $c_2 = 0.8$, $p_m = 0.2$, $V_{\max} = 3$, $n_d = 10$. The optimal settings of parameters for NSGA-II are as follows: $p_c = 0.9$, $p_{m2} = 0.1$, $di_c = 25$, $di_m = 20$.

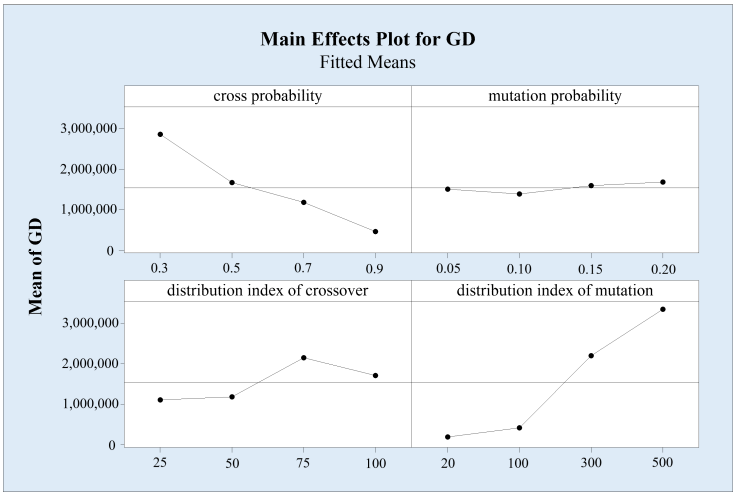


(**a**) Influence of the parameters on generational distance (GD).

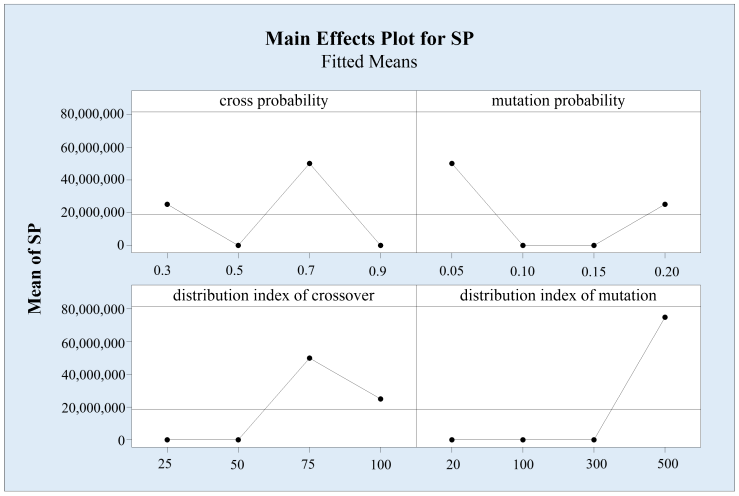


(**b**) Influence of the parameters on spacing (SP).

Figure 6. Influence of the key parameters (MOPSO).



(**a**) Influence of the parameters on GD.



(**b**) Influence of the parameters on SP.

Figure 7. Influence of the key parameters (NSGA-II).

Table 3. Parameters of multi-objective particle swarm optimization (MOPSO).

Parameter	Values				
ω	0.6	0.7	0.8	0.9	1
c_1	0.01	0.05	0.1	0.15	0.2
c_2	0.75	0.8	0.85	0.9	0.95
p_m	0.05	0.1	0.15	0.2	0.25
$V_{\max}$	1	2	3	4	5
n_d	10	20	30	40	50

Table 4. Parameters of NSGA-II.

Parameter	Values			
p_c	0.3	0.5	0.7	0.9
p_{m2}	0.05	0.1	0.15	0.2
di_c	25	50	75	100
di_m	20	100	300	500

4.4. Performance Comparison

We compared our proposed algorithm MOPSO-LS with MOPSO which excluded the local search function completely. We implemented two-opt, three-opt and four-opt in three versions of MOPSO-LS. Besides, we also compare our algorithm with the well-known multi-objective optimization algorithm NSGA-II [19]. We used the optimal parameter values obtained from the DOE for both algorithms. We have generated instances with 15, 30, 50, 75 and 100 jobs randomly. For each size, 10 different instances have been generated, resulting in 50 test instances in total.

To achieve a fair comparison, we imposed the same computational time limit on all algorithms. The time limits were set as follows: 1 s for 15-job instances, 5 s for 30-job instances, 20 s for 50-job instances, 40 s for 75-job instances and 60 s for 100-job instances. To reduce the impact of randomness, each algorithm was run for 20 independent times on each instance and we record the average result. Tables 5 and 6 show the results of average GD and SP for each instance.

From the presented results, the following observations and comments can be made:

1. MOPSO-LS shows excellent performance for almost all instances in terms of GD, which suggests that the solutions obtained by MOPSO-LS are closer to the approximate Pareto front. Since all the computational tests are conducted in limited time, we can judge from the table that the convergence speed of MOPSO-LS is much faster than that of MOPSO and NSGA-II, especially for large-scale instances.
2. For some instances, the NSGA-II achieved better results in terms of SP, meaning that the solutions of NSGA-II are distributed more evenly. However, it should be noted that the NSGA-II solutions were located far from the approximate Pareto front, and in this case, the number of non-dominated solutions tends to be much larger and thus it is easier to find evenly spread solutions. Therefore, even though NSGA-II sometimes obtains a better SP, the inferior GD performance indicates worse solution quality.
3. From the table, we can see that the *k*-opt operator has improved the performance of MOPSO significantly. It is found that three-opt and four-opt are often more powerful than two-opt. However, there is no definite conclusion regarding which one of three-opt and four-opt is more effective (the result appears to be instance-dependent).

To show the results intuitively, we plot the Pareto fronts found by the MOPSO-LS, MOPSO and NSGA-II for an instance with 50 jobs in Figure 8. This figure clearly reveals that our algorithm has obtained much better Pareto fronts than the compared algorithms. It has found solutions with significantly improved total weighted earliness/tardiness and total energy consumption. Noting that earliness/tardiness is an indicator of the manufacturer's profitability, it is safe to conclude that our algorithm has great potential in improving both profitability and energy efficiency for the manufacturing industry.

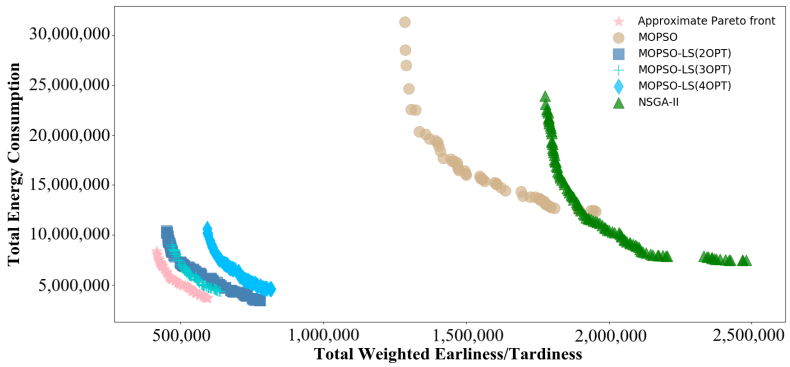


Figure 8. Pareto fronts for an instance with 50 jobs.

Table 5. Comparison of GD.

Size	MOPSO-LS			MOPSO	NSGA-II
	2-opt	3-opt	4-opt		
15	145.84	147.43	154.49	152.77	169.78
	21.53	59.32	77.89	111.21	60.28
	545.36	696.60	625.72	654.69	511.43
	73.52	21.66	21.66	14.04	52.05
	36.27	20.17	41.10	55.28	126.85
	99.61	87.36	73.25	90.70	105.44
	122.1	72.22	68.31	80.01	84.22
	37.66	12.31	13.20	11.20	36.02
	87.21	25.81	11.92	24.52	158.49
	604.48	463.84	569.45	374.33	458.47
30	2.1648×10^4	2.2128×10^4	1.5035×10^4	1.4928×10^4	2.1667×10^4
	8193.22	4345.53	4005.72	4149.82	5625.59
	1.4307×10^4	9405.30	6565.11	1.1149×10^4	1.0322×10^4
	2379.98	1827.37	2465.72	2471.71	2638.53
	1769.88	780.76	510.60	972.319	1395.41
	1315.26	1047.49	1730.21	788.371	2700.17
	449.75	129.74	145.05	104.15	129.403
	1452.09	701.53	916.55	1460.24	2258.18
	2815.46	2088.30	2770.65	3052.56	2385.77
	4152.91	6286.43	5394.63	4709.27	5571.63
50	2.4606×10^5	4.2319×10^4	1.5017×10^5	1.4646×10^5	3.1633×10^5
	1.5841×10^5	3.7033×10^4	1.6799×10^5	7.4834×10^4	2.3097×10^5
	8.9408×10^4	1.5656×10^4	1.5108×10^3	1.9780×10^4	5.7527×10^4
	2.3562×10^5	1.8707×10^5	2.7467×10^5	2.5441×10^5	1.0424×10^6
	5.3550×10^5	2.7078×10^5	1.9046×10^5	1.0054×10^5	2.5288×10^5
	8.5408×10^4	3.2339×10^4	3.5045×10^4	3.4800×10^4	1.2134×10^5
	3.1910×10^6	1.1548×10^6	8.6985×10^5	8.7030×10^5	2.2923×10^6
	8.3748×10^4	3.9804×10^4	4.9790×10^4	1.0186×10^5	2.8793×10^5
	4.0740×10^4	3.2010×10^4	1.4530×10^4	3.7103×10^4	8.6083×10^4
	1.3604×10^5	6.9314×10^4	6.2159×10^4	1.2000×10^5	3.2437×10^5
75	4.5906×10^6	2.1652×10^6	1.4288×10^6	1.2277×10^6	1.8120×10^7
	2.2032×10^6	6.4579×10^5	4.5555×10^5	3.3948×10^5	1.5677×10^6
	2.1880×10^7	1.7321×10^6	4.5183×10^6	1.7686×10^6	1.0907×10^7
	5.5364×10^6	1.1967×10^6	1.6280×10^6	4.5202×10^6	2.6854×10^6
	1.3444×10^7	4.7408×10^6	6.0663×10^6	5.9801×10^6	1.0960×10^7
	2.0739×10^7	1.6263×10^7	8.4651×10^6	5.1385×10^6	1.8489×10^7
	1.4914×10^6	2.8136×10^5	5.6996×10^5	3.2942×10^5	6.8251×10^6
	1.2989×10^7	2.3287×10^6	2.3835×10^6	9.4526×10^5	7.8694×10^6
	2.6751×10^6	5.6586×10^5	6.2739×10^5	7.4984×10^5	3.2544×10^6
	2.1917×10^7	9.9992×10^6	9.0490×10^6	5.8964×10^6	3.4006×10^7
100	9.0453×10^7	5.2840×10^6	1.6884×10^7	1.3406×10^7	1.2505×10^7
	4.2289×10^7	5.2945×10^6	9.3751×10^6	1.5119×10^7	2.6534×10^7
	2.2366×10^9	3.8023×10^8	1.8254×10^9	1.0585×10^9	3.3249×10^9
	1.7385×10^8	2.5784×10^7	2.1698×10^7	8.0410×10^7	3.1927×10^7
	4.1691×10^7	4.6599×10^6	7.6225×10^5	2.4933×10^6	2.3261×10^7
	7.6401×10^7	2.4467×10^7	7.6188×10^7	3.6526×10^7	5.4984×10^8
	1.6481×10^8	1.1901×10^8	1.8326×10^8	1.5697×10^8	1.2890×10^9
	6.0461×10^8	2.5830×10^8	6.7275×10^7	4.4045×10^7	3.4358×10^9
	4.2615×10^8	6.8537×10^7	9.9766×10^7	2.9546×10^7	6.3862×10^8
	5.1326×10^9	7.6756×10^8	5.1069×10^8	1.4917×10^9	9.7009×10^9

Table 6. Comparison of SP.

Size	MOPSO-LS			MOPSO	NSGA-II
	2-opt	3-opt	4-opt		
15	509.19	578.29	584.30	638.47	923.56
	417.51	435.21	555.21	477.75	444.19
	307.09	267.71	366.51	283.58	318.55
	523.81	363.69	363.69	369.30	355.70
	1011.11	1125.7	1081.63	1147.83	1119.74
	306.20	188.57	300.31	232.26	209.33
	357.33	373.69	342.90	414.66	245.61
	315.46	207.31	246.36	222.78	164.90
	514.11	481.24	447.95	453.79	492.89
	517.20	367.61	413.56	686.38	452.984
30	2100.79	1814.01	1698.64	1655.26	1165.11
	6485.56	5272.55	4262.61	4490.44	6924.92
	2165.40	1275.67	1412.01	1486.20	778.504
	2766.03	1390.19	1524.81	1583.83	1093.13
	1284.55	752.24	738.56	859.83	557.49
	1677.26	1111.53	1202.81	1068.90	992.56
	1937.24	875.54	913.88	944.60	709.31
	5533.60	2317.05	2425.59	2530.01	2158.28
	1819.30	934.22	1027.82	944.33	596.92
	4243.48	2499.02	2055.34	1927.15	1582.82
50	1.2823×10^5	4.0749×10^4	6.3635×10^4	5.5120×10^4	6.5409×10^4
	4.8729×10^4	1.8799×10^4	2.2274×10^4	3.4729×10^4	2.1513×10^4
	8.3555×10^3	4.1071×10^3	2.7870×10^3	3.4798×10^3	1.0090×10^4
	1.0361×10^5	4.9401×10^4	5.1010×10^4	4.9039×10^4	1.0937×10^5
	2.0716×10^4	3.2689×10^4	2.0739×10^4	3.3511×10^4	1.6816×10^4
	4.1577×10^4	2.1366×10^4	1.1711×10^4	1.0623×10^4	1.9758×10^4
	4.9620×10^5	3.7802×10^5	2.3289×10^5	1.8961×10^5	5.9416×10^5
	6.8611×10^4	3.2026×10^4	1.3416×10^4	3.1120×10^4	4.2603×10^4
	2.8763×10^4	1.2816×10^4	1.0761×10^4	1.2676×10^4	1.1313×10^4
	1.1180×10^5	3.2213×10^4	2.4039×10^4	7.6251×10^4	6.3371×10^4
75	8.2139×10^5	3.1320×10^5	2.1188×10^5	1.8545×10^5	1.7680×10^6
	2.4810×10^5	6.7156×10^4	5.8004×10^4	1.5406×10^5	1.7449×10^5
	2.7678×10^6	2.4515×10^5	1.4072×10^6	1.3681×10^6	1.7745×10^6
	5.0937×10^5	2.8875×10^5	2.1010×10^5	1.1063×10^6	3.2958×10^5
	2.3024×10^6	9.0459×10^5	1.2373×10^6	1.1083×10^6	1.2357×10^6
	1.6779×10^6	4.4779×10^5	1.7220×10^6	4.9882×10^5	1.6083×10^6
	2.8209×10^5	7.5955×10^4	8.7421×10^4	6.9395×10^4	6.0464×10^5
	3.3010×10^6	2.7282×10^5	7.2861×10^5	2.8029×10^5	6.7773×10^5
	1.1634×10^5	5.0719×10^4	6.6612×10^4	6.8837×10^4	1.8110×10^5
	5.1115×10^6	3.3516×10^6	2.2598×10^6	2.1649×10^6	5.4228×10^6
100	3.0893×10^6	1.0378×10^6	2.8150×10^6	4.8964×10^5	3.3332×10^6
	3.3866×10^6	3.8091×10^5	4.1121×10^5	1.1611×10^6	2.7173×10^6
	3.5322×10^6	1.8074×10^7	1.8546×10^7	8.8837×10^5	1.2989×10^6
	4.0513×10^6	2.2268×10^6	6.5448×10^6	5.2987×10^7	4.2246×10^6
	4.8506×10^6	4.2216×10^5	2.1611×10^5	1.1851×10^5	3.9152×10^6
	1.4778×10^7	3.1322×10^6	1.0099×10^7	3.2020×10^6	3.3094×10^6
	1.5716×10^7	1.4263×10^7	3.7640×10^7	3.7561×10^6	3.5944×10^6
	1.7767×10^7	8.0118×10^6	3.8425×10^7	7.1203×10^6	4.9552×10^6
	1.7079×10^7	3.7224×10^6	9.1961×10^6	4.0269×10^6	1.8599×10^6
	2.1737×10^6	9.5046×10^6	3.3712×10^6	1.7927×10^7	1.1332×10^6

5. Conclusions

This paper investigates the single-machine scheduling problem with deterioration to minimize two objectives simultaneously, i.e., the total weighed earliness/tardiness and the total energy consumption. The first objective reflects the just-in-time management philosophy and the second objective reflects the motivation to pursue sustainable and green manufacturing.

We propose a hybrid meta-heuristic algorithm called MOPSO-LS which enhances MOPSO by a dedicated local search procedure to solve the problem. The local search strategy is built on the idea of *k*-opt neighborhood operators and has been adapted for the scheduling problem. To evaluate the effectiveness and efficiency of the proposed algorithm, computational experiments are conducted and comparisons are made between MOPSO-LS, MOPSO and NSGA-II. The results show that the MOPSO-LS has faster convergence speed and better solution quality compared to the other two algorithms. Besides, it is shown that the local search procedure based on 2-opt, 3-opt and 4-opt have improved the search ability of MOPSO significantly.

In future research, we will focus on more advanced energy models and more sophisticated local search strategies. The energy consumption of machines in different states and the different power rates under adjustable processing speeds will be taken into consideration. This will result in a more complicated scheduling problem with additional decision variables, which makes it necessary to devise an enhanced local search scheme to promote the search efficiency of meta-heuristic algorithms. The *k*-opt neighborhood operators could be combined with more complex moves to construct an adaptive large neighborhood search policy.

Author Contributions: Conceptualization, Y.L. and R.Z.; methodology, Y.L. and R.Z.; software, X.L.; validation, Y.L., X.L. and R.Z.; formal analysis, Y.L.; investigation, Y.L. and X.L.; resources, R.Z.; data curation, X.L.; writing—original draft preparation, Y.L.; writing—review and editing, R.Z.; visualization, Y.L.; supervision, R.Z.; project administration, R.Z.; funding acquisition, R.Z.

Funding: This research was funded by the Natural Science Foundation of China (grant number U1660202).

Acknowledgments: The authors are grateful to the anonymous reviewers for constructive comments which have helped to improve this paper.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Wang, S.; Liu, M.; Chu, F.; Chu, C. Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration. *J. Clean. Prod.* **2016**, *137*, 1205–1215. [CrossRef]
2. Wang, H.; Alidaee, B. Effective heuristic for large-scale unrelated parallel machines scheduling problems. *Omega* **2019**, *83*, 261–274. [CrossRef]
3. Plitsos, S.; Repoussis, P.P.; Mourtos, I.; Tarantilis, C.D. Energy-aware decision support for production scheduling. *Decis. Support Syst.* **2017**, *93*, 88–97. [CrossRef]
4. Aghelinejad, M.; Ouazene, Y.; Yalaoui, A. Complexity analysis of energy-efficient single machine scheduling problems. *Oper. Res. Perspect.* **2019**, *6*, 100105. [CrossRef]
5. Zhang, R.; Chiong, R. Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *J. Clean. Prod.* **2016**, *112*, 3361–3375. [CrossRef]
6. Wu, X.; Shen, X.; Li, C. The Flexible Job-Shop Scheduling Problem Considering Deterioration Effect and Energy Consumption Simultaneously. *Comput. Ind. Eng.* **2019**, *135*, 1004–1024. [CrossRef]
7. Liao, X.; Zhang, R.; Chiong, R. Multi-objective optimization of single machine scheduling with energy consumption constraints. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–8.
8. Módos, I.; Šůcha, P.; Hanzálek, Z. Algorithms for robust production scheduling with energy consumption limits. *Comput. Ind. Eng.* **2017**, *112*, 391–408. [CrossRef]

9. Khorshidian, H.; Javadian, N.; Zandieh, M.; Rezaeian, J.; Rahmani, K. A genetic algorithm for JIT single machine scheduling with preemption and machine idle time. *Expert Syst. Appl.* **2011**, *38*, 7911–7918. [CrossRef]
10. Behnamian, J. A parallel competitive colonial algorithm for JIT flowshop scheduling. *J. Comput. Sci.* **2014**, *5*, 777–783. [CrossRef]
11. Wang, J.B.; Xia, Z.Q. Flow shop scheduling with deteriorating jobs under dominating machines. *Omega* **2006**, *34*, 327–336. [CrossRef]
12. Wang, J.B.; Ng, C.; Cheng, T.E. Single-machine scheduling with deteriorating jobs under a series–parallel graph constraint. *Comput. Oper. Res.* **2008**, *35*, 2684–2693. [CrossRef]
13. Wang, J.B.; Liu, L.L. Two-machine flow shop problem with effects of deterioration and learning. *Comput. Ind. Eng.* **2009**, *57*, 1114–1121. [CrossRef]
14. Lenstra, J.K.; Kan, A.R.; Brucker, P. Complexity of machine scheduling problems. *Ann. Discrete. Math.* **1977,** *1*, pp. 343–362.
15. Coello Coello, C.A.; Pulido, G.T.; Lechuga, M.S. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. [CrossRef]
16. Van Veldhuizen, D.A.; Lamont, G.B. Evolutionary Computation and Convergence to a Pareto Front. 1998. Available online: https://pdfs.semanticscholar.org/f329/eb18a4549daa83fae28043d19b83fe8356fa.pdf (accessed on 1 August 2019).
17. Schott, J.R. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*; Technical Report; Air Force Inst of Tech Wright-Patterson AFB OH: Dayton, OH, USA, 1995.
18. Taguchi, G.; Phadke, M.S. Quality engineering through design optimization. In *Quality Control, Robust Design, and the Taguchi Method*; Springer: Boston, MA, USA, 1989; pp. 77–96.
19. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]