sustainability

MDPI

Article

Technology for Production Scheduling of Jobs for Open Innovation and Sustainability with Fixed Processing Property on Parallel Machines

Sang-Oh Shim [1] and KyungBae Park [2,*]

1 Department of Business Administration and Accounting, Hanbat National University, 16-1 Deokmyung-Dong, Yuseong-Gu, Daejeon 34158, Korea; soshim@hanbat.ac.kr

2 Department of Business Administration, Sangji University, 660 Woosan-Dong, Wonju-Si, Kangwon 26339, Korea

* Correspondence: kbpark@sangji.ac.kr; Tel.: +82-10-2896-0745

Academic Editors: Tan Yigitcanlar and Marc A. Rosen

Received: 29 June 2016; Accepted: 30 August 2016; Published: 7 September 2016

Abstract: In this paper, a technology for production scheduling is addressed for the sustainability and open innovation in a manufacturing business. Methodologies for scheduling jobs on parallel machines with the fixed processing property are devised. The fixed processing property, in which a group of specific jobs can be processed on the predetermined machine, can be found in most manufacturing systems due to the quality issues. Usually, even though parallel machines can process various types of jobs, the fixed processing is preferred as to not deteriorate products' quality in real manufacturing systems. To minimize makespan of jobs, which is defined as the final completion time of all jobs, technology for production scheduling is developed. Several heuristic algorithms are devised for solving the problem and to evaluate performance of the suggested algorithms, a series of computational experiments is performed. Results show that better solutions are obtained by the suggested algorithms in a reasonable amount of computation time. That is, if the proposed technology is applied to the production scheduling system of a real manufacturing business, it can be expected that quantity and quality of the product will be enhanced since they are influenced by the production scheduling.

Keywords: technology; open innovation; manufacturing; sustainability; heuristics; production scheduling; parallel machines

1. Introduction

Nowadays, in the field of operation management and supply value chain of manufacturing business, open innovation and sustainability are getting more necessary [1,2]. For example, in view of higher level hierarchy of manufacturing business, i.e., policy and strategy making, issues for using Big data in information technology (IT) products industries are discussed [1] and the relationship between total quality management (TQM) practices and innovative performances is addressed [2]. On the other hand, in view of a lower level, i.e., operational level, smart factory, also called "industry 4.0", is currently raised for the sustainability.

It is usually defined as the automation manufacturing system composed of cyber-physical systems (CPS), internet of things (IoT), and cloud computing [3]. Through the smart factory, it can be expected that customization of products, mass production, manufacturing flexibility, and improvement of products' quality are possible. In addition to external components of the smart factory, i.e., CPS, IoT, and cloud computing, internally, one of the most important thing is intelligent production scheduling, i.e., how to schedule jobs effectively and efficiently to maximize production quantity and quality, customer satisfaction, and so on. In this paper, a technology for production scheduling of jobs in manufacturing systems is addressed since sustainability in manufacturing operation scheduling is discussed lately [4,5]. In those studies, scheduling is one of the very important key factors at the operational level for the sustainability since it influences production quantity, quality, and customer satisfaction. Also, according to the result in scheduling, resource consumption, energy efficiency, and emissions can be effected.

We proposed several heuristic algorithms for scheduling jobs on parallel machines with the fixed processing property in which a group of specific jobs can be processed on the predetermined machine. This property can be found at most manufacturing systems which produce state-of-the-art technology goods, such as semiconductors, liquid crystal displays, and so on, due to quality issues. Usually, even though it is known that parallel machines can process various types of jobs, this property is preferred as to not deteriorate products' quality in real manufacturing systems. Also, in this workstation, when changing processes of different groups of jobs, operations for changing types of groups, called setup, are necessary.

In semiconductor fabrication, most of the workstations, such as chemical vapor deposition (CVD) and ion implant processing (IIP) are composed of parallel fixed processing machines, each of which can process only a set of groups that is preferred to the predetermined machine. Also, to process jobs, since changes of process that specify processing conditions are required for each group of jobs, a setup occurs when changing to a different group of jobs.

In real manufacturing, setup operations are not preferred due to quality issues and their duration, that is, if there are jobs of changing processes on the machines, products might be damaged due to the previous process. Also, if setup time is relatively long, compared with the processing time of jobs, managers want to reduce the number of setup operations. Hence, these workstations, i.e., CVD and IIP, have fixed processing property for high utilization of the machine and quality of products even though all machines are parallel so that they can process all product groups. Once a group of jobs is processed on a machine, the same group of jobs is wanted to be processed on the same machine if there is a group of jobs. In this paper, we assume that the fixed processing property is known, that is, we know that each machine has predetermined processes.

Usually to increase the production quantity, in the manufacturing firm, makespan is used as a performance measure. It is defined as a completion time of last completed job, in this paper, technology for production scheduling is developed, that is, several heuristic algorithms are devised for solving the problem considered here. The remainder of this paper is organized as follows. In the next section, the literature review is suggested. In Section 3, several heuristics algorithms are devised and in Section 4, to evaluate performance of the suggested algorithms, a series of computational experiments is performed on a number of randomly generated test problems and the results are shown. In the final section, we summarize this paper.

2. Literature Reviews

A number of studies about scheduling technology to enhance the ability of production in manufacturing systems have been done [6–14]. On the other hand, studies for parallel machines with fixed processing property are very rare.

Few studies on scheduling problems for parallel fixed processing property are addressed. Lee and Kim [15] considered a scheduling technique in a flexible manufacturing system in which each job should be assigned to one of the predetermined group of machines. Glass et al. [16] considered the scheduling problem on m-machines with the fixed processing property, in which only one machine must process the predetermined jobs, to minimize the makespan. On the other hand, in this study, each group of job can be processed on several fixed machines, therefore, jobs in the same group can be split to process them on different machines if they have the same fixed processing property. Also, Goemans [17] deals with scheduling problem on three machines with fixed processing property to minimize the makespan and develop a near optimal algorithm. Several dispatching rules, considering fixed processing property are proposed by Wu et al. [18].

For the problem of minimizing the makespan on the parallel shops—that is a general case of the problem considered here—lots of studies are done. For the representative examples, by assigning the longest processing time job to a machine with the earliest possible staring time, dispatching rule, named LPT (Longest Processing Time), is introduced [19]. For the same problem, Coffman et al. [20] developed the scheduling algorithm, i.e., the MULTIFIT algorithm, to solve the bin packing problems. Based on the LPT and MULTIFIT rules, later studies for minimizing makespan in parallel machine shops are developed. On the other hand, heuristic methods and a neural network are used by Lee et al. [21] and Akyol and Bayhan [22], respectively.

Lots of scheduling problems on parallel machine shops with setup operations are studied in previous studies. Wittrock [23] and So [24] dealt with the scheduling problem with two types of setup, minor and major. Also, Xing and Zhang [25] developed a heuristic algorithm in which a group of job can be split and processed on more than two parallel machines by using results of Monma and Potts [26].

3. Technology for Production Scheduling

As described above, to minimize the completion time of the last job, i.e., makespan of jobs, this study addresses the scheduling problem in which several groups of jobs are processed on the predetermined parallel machine which has a fixed processing property while considering setup operations between different groups of jobs. Jobs are already grouped according to those processes, i.e., processing condition, and processing (and setup) time is same if they are in same group. We assume that it is already known that process information is assigned to each machine, so that each machine should complete jobs of the group specified to the assigned process.

Setup operation is sequence independent and note that setup time is relatively longer than processing time. Also, we assume that all jobs to be scheduled are waiting at the time of scheduling; preemption of job is not allowed.

In the parallel-machine shop, since it is known that the schedule in which jobs of same group are consecutively sequenced is dominant for the regular measures including makespan [27], we do not consider the sequence in which jobs of same group are sequenced separately on the same machine. Also, since the makespan is defined as the completion time of the last job, sequence of jobs on each machine is not important and only assignment of jobs should be considered. Therefore, this scheduling problem can be transformed to the one of assigning jobs of groups to the machines where these jobs of same group are consecutively sequenced.

In this paper, we present several heuristics to develop the technology for scheduling jobs on the parallel machines with the fixed processing property. First, we develop the scheduling algorithm for making the initial schedule by assigning jobs of group to the fixed processing machine.

3.1. Developing the Initial Schedule by Considering Subproblems

To develop the initial schedule, we divide the original one into independent subproblems. Each subproblem is solved independently and then we combine the results of all subproblems so that the initial schedule is obtained. To solve each one, various rules are developed and algorithms for combining each result are suggested.

To divide the original problem into subproblems, by assigning each group of jobs to only one subproblem, it can be transformed to a bin packing problem with identical size. First, we figure out which groups can be processed on only one machine. For example, if group 1 and group 2 are processed on only machine 1 due to a fixed processing property, then a subproblem is two groups on a single machine scheduling problem. Then, we figure out which groups can be processed on only two machines. For example, if groups 3 and 4 can be processed on machine 1 and 2, then a second subproblem can be a two-group scheduling problem with two machines. We repeat this procedure until all groups are assigned to subproblems. Therefore, the number of subproblems cannot be exceeded to the number of group of jobs.

To solve each subproblem, i.e., *n*-job, *m*-machine scheduling problem to minimize makespan, obtained in the above procedure, two methods are devised. In the first method based on the results of Graham [19], a job of a group with the highest priority is assigned to the machine with the earliest starting time, i.e., least workload. Hence, we use four priority rules for choosing a job, (i) shortest processing time; (ii) longest processing time; (iii) number of remaining unscheduled jobs of a group; and (iv) total processing time of remaining unscheduled jobs of a group.

Since the subproblems can be transformed to bin packing problems, in the second method based on the results of Coffman et al. [20], one is modified from the MULTIFIT algorithm is developed. First, we compute minimum and maximum bin size considering setup requirements. Minimum bin size of each subproblem, named as MIN in this study, is computed with the assumption that processing and setup operations time are assigned evenly to each machine. Therefore, in each subproblem, MIN is obtained as $\sum_{j \in N} p_j / E + s \times \max(0, |G/E|)$ where p_j is a processing time of a job j, N is a set of jobs, E is a number of machine, s is a time of setup operation, and G is a number of group of jobs, respectively. On the other hand, maximum bin size of each subproblem, named as MAX, can be computed suppose that all jobs are assigned to only one machine. Hence, MAX is calculated as $\sum_{j \in N} p_j + s \times G$. The procedure of the second method is as follows:

(i) Set initial size of each bin, i.e., machine, as (MAX + MIN)/2.
(ii) Select a job using by priority rules described in the first method. If there is no more job to be assigned, then the current size of bin and allocation are the initial makespan and schedules, respectively, stop.
(iii) Select a machine with the earliest possible start time among available machines of which the workload allocated does not exceed the current size of bin. If there is no available machine, set MIN as the current size of bin and go to step (i).
(iv) If current size of bin is less than the current workload of the machine, then assign the job to the machine (if setup operation is required, then add the time of setup operation) and update the machine workload. Otherwise go to step (ii).

3.2. Developing the Initial Schedule by Original Problem

In the above algorithms, the problem is decomposed into the several problems to obtain the initial schedule. On the other hand, in this study we solve the original problem directly by suggesting two methods. In the first method, we assign the all jobs to the fixed processing machine evenly. That is, if there is a group of jobs to be allocated to only one machine, then all jobs of this group are assigned to the machine. Also, if there is a group to be allocated to three machines by the fixed processing property, then all jobs are split into three sub-jobs and are assigned to those machines. After assigning all groups of jobs by considering setup, for the all split groups, the sub-jobs with the smallest processing time in the machines with the maximum workload are moved into the machine with the smallest workload among ones which has same group of sub-jobs. If there is no improvement for makespan during moving procedure, we select the next sub-job. This moving procedure is repeated until all split groups are checked. Then, the initial schedule is obtained.

In the second method, based on the dynamic priority rules for groups and machines developed here we propose the modified MULTIFIT algorithm. We set P_{ge} as priority for group of jobs, g, and machine, e. For group g and machine e, initial P_{ge} is set to 1 divided by the number of the fixed processing machines, which can process group g and set to 0 for the machine, which cannot process group g. We select the group and machine with the highest P_{ge} and a job of the selected group is assigned to the selected machine and update workload and dynamic priorities.

Similarly, as described in the above MULTIFIT algorithm, minimum bin size (MIN) and maximum bin size (MAX) for the original problem are computed as following. MIN is obtained as $\sum_{j \in N} p_j / E + s \times \max(0, |G/E|)$ where P_j is a processing time of a job j, N is a set of all jobs, E is the number of all machines, s is a time of setup operation and G is a number of all groups, respectively. Also, MAX is calculated as $\sum_{j \in N} p_j + s \times G$. The procedure of the second method is as follows:

(i) Set initial size of each bin, i.e., machine, as (MAX + MIN)/2.
(ii) Select a group and a machine with the highest priority using by dynamic priority rules, P_{ge}. In case of ties, select the group with longest processing time and a machine with the smallest estimated workload. The estimated workload is computed as the sum of the current workload and workload (multiplied by P_{ge}) obtained by assuming that remaining jobs of other groups except for the selected group are assigned to the machine. If there is no more job to be assigned, then the current size of bin and allocation are the initial makespan and schedules, respectively, stop.
(iii) If the allocated workload exceeds the current size of bin (if setup operation is required, then add the time of setup operation), set MIN as the current size of bin and go to step (i). Otherwise, assign the job to the machine and update the machine workload and P_{ge} for all groups and machines as follows. For selected group and non-selected machines, new P_{ge} is set to 0, for non-selected groups and selected machine, new P_{ge} is computed as (1 − current workload/current bin size) × current P_{ge}, for non-selected groups and machines, new P_{ge} is calculated as (1 − current workload/(number of non-selected machines − 1) × current bin size) × current P_{ge}. Go to step (ii).

3.3. Construction Method for Improving the Initial Schedule

By using the four algorithms suggested in Sections 3.1 and 3.2, the initial complete schedules can be obtained and these are improved by a construction method. The improvement is done by moving jobs between two machines. The procedure to improve the initial schedule is as follows:

(i) From the initial schedule, make U_e as the set of machines in descending order of workload.
(ii) Select the first machine in U_e and denote the machine as FROMEQP.
(iii) Figure out the scheduled groups of jobs on the machine in descending order of their processing time and make U_g as the set of these groups.
(iv) Select the first group in U_g and denote the group as MOVEGRP. If there is no selected group, delete the FROMEQP in U_e and go to step (ii).
(v) Find another machine which can process the MOVEGRP by the fixed processing property. If there is no machine, then delete MOVEGRP in U_g and go to step (iv). Otherwise, make T_e as set of them in ascending order of workload.
(vi) Select the first machine in T_e and denote this machine as TOEQP.
(vii) By using property suggested in Kim [28], compare the workloads of the selected machine, TOEQP, with the one of the machine, FROMEQP, assuming that a job of the selected group, MOVEGRP, is moved from FROMEQP to TOEQP (if setup operation is required, then apply setup time).
(viii) If there is an improvement for makespan measure, then move the job and repeat previous step until no more improvement is done. Otherwise, delete TOEQP in T_e and go to step (vi). If there is no element in T_e, then delete the selected group in U_g and go to step (iv).
(ix) If there is no more improvement on the machine, FROMEQP, then delete it in U_e and go to step (ii).
(x) Repeat this procedure until there is no element in *Ue*.

4. Computational Experiments

In order to evaluate the performance of the proposed heuristic algorithms, a series of computational tests on randomly generated problem instances are performed reflecting real manufacturing situation. All the algorithms presented in this study were coded in C programming language, and computational tests were performed on a personal computer with an Intel Core i3-4030U processor operating at 1.9 GHz clock speed.

Four heuristic algorithms introduced in Section 3, which are named as ALG1, ALG2, ALG3, and ALG4, respectively, are tested for the evaluation. The ALG1, suggested in Section 3.1, is the one which solves subproblems by using four priority rules separately and uses construction methods (Section 3.3) for improvement. ALG2, suggested in Section 3.1, is the one which solves subproblems by using modified MULTIFIT algorithm separately and uses construction methods (Section 3.3) for improvement, ALG3, suggested in Section 3.2, is the one which solves original problem by moving sub-jobs considering makespan and uses construction methods (Section 3.3) for improvement and ALG4, suggested in Section 3.3, is the one which solves original problem by modified MULTIFIT algorithm and uses construction methods (Section 3.3) for improvement. We compare them with an existing method that has been used in a real manufacturing systems. In this method, when a machine becomes available, a job in the same group is selected to avoid setup operation, that is, continuing same group operation, and assigned to the machine (If there is no job of same group, a job is selected randomly and setup occurs).

As the performance measures, the number of best solution (NBS) found by each algorithm and percentage reduction (PR), defined as $100 \times (1 - M_a/M_r)$, where M_a is the makespan obtained by algorithm a, M_r is the makespan obtained from the method currently used in real manufacturing system, are used. If the performance of the suggested algorithm, i.e., M_a, is better than the one of the current method used in real situation, i.e., M_r, PR can be expected to be close to 100%.

For the tests, we generate 900 test instances randomly, 50 problems for each of all 18 combinations of three levels for the number of jobs (30, 60, and 90), three levels for the number of groups (3, 6, and 9), and two levels for the number of machines (5 and 10). Processing times of a job are generated from discrete uniform distribution with range [5,25]. Setup time is set to 30 for all groups. For the fixed processing property, the number of groups that each machine can process is generated from uniform distribution with range [1, X], where X is a number of group and each group are determined arbitrarily.

The overall results of the computational experiments are shown in Table 1. All of the suggested algorithm perform better than the method used in real manufacturing system since the all average percentage reductions are greater than zero. Additionally, all average PR values are more than 10%, it implies that makespan in the real manufacturing system can be reduced significantly by using the suggested algorithms. Also, all solutions are obtained in less than 0.1 s in terms of computational time.

Regardless of the various sizes of jobs, groups, and machines, the better performance of the algorithms are shown consistently, therefore, it means that the suggested algorithms are robust so that these are suitable for real manufacturing systems. Although the results without applying the construction methods for improvement (Section 3.3) are not shown in the results, better performance was found at most of the problems with the range of PR from 0% to 10%. In few problems, the PR values without applying the construction methods for improvement were negative. However, after applying the construction methods for improvement, solutions of all the problems were improved significantly.

Even if all the algorithms show better performance, it seems that ALG3 works better than others, i.e., ALG1, ALG2, and ALG4 due to the number of best solution obtained by ALG3. That is, the algorithm in which the initial schedule obtained by solving the original problem is improved by the construction methods is good at solving this scheduling problem with the fixed processing property. However, as it can be expected from the table, there are no significant differences on the PR values between ALG3 and ALG1, ALG3, and ALG4 statistically (There is significant difference only between ALG3 and ALG1).

Table 1. Overall results of the suggested algorithms.

Number of Jobs	Number of Groups	Number of Machines	ALG1		ALG2		ALG3		ALG4	
30	3	5	13.8 [†]	14 [††]	12.1	11	22.3	29	20.1	25
		10	15.5	12	13.2	11	20.5	25	19.8	20
	6	5	20.1	16	11.5	11	19.8	15	22.3	18
		10	18.1	17	10.9	10	22.3	21	18.4	19
	9	5	18.3	16	13.2	12	18.5	20	19.6	21
		10	19.4	20	14.5	15	19.5	20	21.7	22
60	3	5	15.4	14	17.2	16	21.4	19	22.7	25
		10	22.3	25	11.4	10	20.9	23	20.5	22
	6	5	14.3	15	13.4	12	23.4	20	22.1	21
		10	16.4	12	14.2	12	22.5	25	25.6	30
	9	5	14.1	13	13.0	14	25.1	21	24.1	28
		10	19.4	20	12.9	14	22.3	26	20.7	22
90	3	5	17.5	19	14.1	16	20.8	25	18.4	20
		10	17.2	18	15.8	16	27.4	29	19.4	22
	6	5	18.5	19	13.2	12	25.3	22	20.1	19
		10	18.2	19	14.6	11	22.2	21	15.0	16
	9	5	20.2	21	16.2	10	23.0	20	18.9	13
		10	22.3	28	13.9	17	25.9	34	23.5	26

[†] Percentage reduction of heuristic algorithm to the result of practice; [††] Number of cases that heuristic algorithm found the best solution.

To evaluate the absolute performance of the suggested algorithms, the best solution among the ones obtained from the algorithms is compared with optimal one. To obtain the optimal solution, we formulate the problem considered here as a mixed integer programming. Here, notation used in the mixed integer programming is suggested.

p_g	processing time of a job of group g
s	sequence independent setup time
e	index for machines
g	index for groups
J_g	set of jobs of group g
E_g	set of machines on which jobs of group g can be processed, i.e., fixed processing property
x_{gje}	decision variable, which is equal to 1 if job j of group g is assigned to machine e, otherwise, 0, for $e \in E_g$
y_{ge}	decision variable, which is equal to 1 if jobs of group g are assigned to machine e, otherwise, 0, for $e \in E_g$

Also, the mixed integer programming formulation is as follows:

$$\text{Minimize} \quad C_{\max} \tag{1}$$

$$\text{Subject to} \quad x_{gje} \le y_{ge} \qquad \forall g, \forall e, \forall j \in J_g \tag{2}$$

$$\sum_g s y_{ge} + \sum_g \sum_{\substack{j \in \in \\ g}} x_{gje} p_g \le C_{\max} \qquad \forall e \tag{3}$$

$$\sum_e x_{gje} = 1 \qquad \forall g, \forall j \in J_g \tag{4}$$

$$x_{gje} \in \{0, 1\} \qquad \forall g, \forall e, \forall j \in J_g \tag{5}$$

$$y_{ge} \in \{0, 1\} \qquad \forall g, \forall e \tag{6}$$

Constraint (1) and (2) represent that the performance measure of the problem is makespan and setup operation is needed, respectively. Also, Constraint (3) shows the workload (completion time) on each machine, and constraint (4) introduces that a job should be assigned to only one machine. Last two constraints, (5) and (6), describe that decision variables are binary.

Since the mixed integer programming could not solve the large-sized problem in an acceptable amount of computation time, the best solution among the ones obtained from the algorithms is compared with optimal one for the small sized problem. The optimal solution is obtained by using commercial mathematical programming tool, ILOG CPLEX version 10.0. We use percentage gap as the performance measure, which is computed as $100 \times (M_b/M_o - 1)$ where M_b is the best makespan obtained by the suggested algorithms and M_o is the optimal makespan obtained by solving the mixed integer problem. We test to obtain the optimal solution in 600 s (If the optimal solution cannot be obtained in 600 s, we stop the program and obtain the current solution). We generate 20 test instances randomly, 10 problems for 2 combinations of 30 jobs, 5 machines and 3 and 6 for the number of groups. Other data are generated in same way described above.

In Table 2, the test results in order to evaluate the absolute performance of the suggested algorithms are shown. Since the percentage difference from the optimal solution is less than 4%, it argues that the solution quality is very good, even though we test only a small-sized problem due to the limitation of computation time. Also, it can be seen that the computation time to obtain the solution is only around 0.04 s in the suggested algorithms as well as the solution quality is near optimal (less than 4 %). Therefore, the proposed algorithms show very good performance in terms of computation time and solution quality.

Table 2. Result of comparison tests over optimal solutions.

Number of Jobs	Number of Groups	Number of Machines	APG [†]	ACTH [††]	ACTO [†††]
30	3	5	3.85	0.03	485.1
30	6	5	3.25	0.04	452.5

[†] Average percentage gap between heuristic solutions and optimal solutions; [††] Average computation time to obtain the best solution among the heuristic algorithms, in seconds; [†††] Average computation time to obtain the optimal solution, in seconds.

5. Conclusions

The necessities and importance of sustainability and open innovation are growing in over the world currently [29–31] and in the operational level of the manufacturing company, they are becoming more important [1–5,14]. One of the efforts for sustainability is an intelligent automation system, called a smart factory, and we considered a technology for production scheduling of jobs in manufacturing systems for it. Since scheduling results effect the production quantity, quality, and customer satisfaction, scheduling jobs effectively and efficiently is one key factor for sustainability.

In this research, we dealt with a scheduling problem on parallel shops with the fixed processing property to minimize the makespan of jobs. The fixed processing property is that a group of specific jobs should be processed on the predetermined machine. Due to the quality of products or managerial convenience, this property can be found at the most manufacturing systems, so that it is very important to schedule jobs while considering it. Four heuristic algorithms, which are based on the priority rules and MULTIFIT algorithms, and construction improvement methods, are devised for solving the problem. These methods give better results in a reasonable amount of computation time as well as in terms of solution quality.

If the proposed technology is applied to the real production scheduling system, production quantity and quality are enhanced, hence customer satisfaction is increased without no additional investment. Consequently, it is necessary to develop efficient and effective scheduling methodologies for the sustainability of the operational level in the manufacturing business.

In further research, we can extend this study in several ways. For example, one may consider a general case of the problem in which setup operation is not sequence independent so that setup times are different from sequence of group of jobs. Also, it may address the problem with a case of dynamic arrival of jobs so that all jobs are not available at the time of scheduling.

Acknowledgments: This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2012S1A5A8024194).

Author Contributions: All authors significantly contributed to the scientific study and writing. Sang-Oh Shim contributes to the overall idea, algorithms, and writing of the manuscript; KyungBae Park contributes to the detailed writing, ideas, and discussions on open innovation and sustainability of operation system and manufacturing firms, as well as preparation of publishing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baek, H.; Park, S. Sustainable development plan for Korea through expansion of green IT: Policy issues for the effective utilization of big data. *Sustainability* **2015**, *7*, 1308–1328. [CrossRef]
2. Yusr, M.M. Innovation capability and its role in enhancing the relationship between TQM practices and innovation performance. *J. Open Innov. Technol. Mark. Complex.* **2016**, *2*. [CrossRef]
3. Lee, J.; Bagheri, B.; Kao, H.A. Recent advances and trends of cyber-physical systems and big data analytics in industrial informatics. *Int. Conf. Ind. Inform.* **2014**. [CrossRef]
4. Kleindorfer, P.R.; Singhal, K.; Wassenhove, L.N.V. Sustainable operations management. *Prod. Oper. Manag.* **2005**, *14*, 482–492. [CrossRef]
5. Giret, A.; Trentesaux, D.; Prabhu, V. Sustainability in manufacturing operations scheduling: A state of the art review. *J. Manuf. Syst.* **2015**, *37*, 126–140. [CrossRef]
6. Wein, L.W. Scheduling semiconductor wafer fabrication. *IEEE. Trans. Semicond. Manuf.* **1988**, *1*, 115–130. [CrossRef]
7. Lu, S.C.H.; Ramaswamy, D.; Kumar, P.R. Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plant. *IEEE. Trans. Semicond. Manuf.* **1994**, *7*, 374–388. [CrossRef]
8. Uzsoy, R. Scheduling a single batch processing machine with non-identical job sizes. *Int. J. Prod. Res.* **1994**, *32*, 1615–1635. [CrossRef]
9. Kim, Y.D.; Lee, D.H.; Kim, J.U.; Roh, H.K. A simulation study on lot release control, mask scheduling, and batch scheduling in semiconductor wafer fabrication facilities. *J. Manuf. Syst.* **1998**, *17*, 107–117.
10. Lee, Y.H.; Park, J.; Kim, S. Experimental study on input and bottleneck scheduling for a semiconductor fabrication line. *IIE Trans.* **2002**, *34*, 179–190. [CrossRef]
11. Jain, V.; Swarnkar, R.; Tiwari, M.K. Modeling and analysis of wafer fabrication scheduling via generalized stochastic Petri net and simulated annealing. *Int. J. Prod. Res.* **2003**, *41*, 3501–3527. [CrossRef]
12. Kim, Y.D.; Shim, S.O.; Choi, B.; Hwang, H. Simplification methods for accelerating simulation-based real-time scheduling in a semiconductor wafer fabrication facility. *IEEE. Trans. Semicond. Manuf.* **2003**, *16*, 290–298.
13. Mason, S.J.; Jin, S.; Wessels, C.M. Rescheduling strategies for minimizing total weighted tardiness in complex job shops. *Int. J. Prod. Res.* **2004**, *42*, 613–628. [CrossRef]
14. Tong, Y.; Li, J.; Li, S.; Li, D. Research on energy-saving production scheduling based on a clustering algorithm for a forging enterprise. *Sustainability* **2016**, *8*, 136. [CrossRef]
15. Lee, D.H.; Kim, Y.D. Scheduling algorithms for flexible manufacturing systems with partially grouped machines. *J. Manuf. Syst.* **1999**, *18*, 301–309. [CrossRef]
16. Glass, C.A.; Shafransky, Y.M.; Strusevich, V.A. Scheduling for parallel dedicated machines with a single server. *Nav. Res. Logist.* **2000**, *47*, 304–328. [CrossRef]
17. Goemans, M.X. An approximation algorithm for scheduling on three dedicated machines. *Discret. Appl. Math.* **1995**, *61*, 49–59. [CrossRef]
18. Wu, M.C.; Huang, Y.L.; Chang, Y.C.; Yang, K.F. Dispatching in semiconductor fabs with machine-dedication features. *Int. J. Adv. Manuf. Technol.* **2006**, *28*, 978–984. [CrossRef]
19. Graham, R.L. Bounds on multiprocessor timing anomalies. *SIAM J. Appl. Math.* **1969**, *17*, 416–429. [CrossRef]
20. Coffman, E.G.; Garey, M.R.; Johnson, D.S. An application of bin-packing to multi-processor scheduling. *SIAM J. Comput.* **1978**, *7*, 1–17. [CrossRef]

21. Lee, W.C.; Wu, C.C.; Chen, P. A simulated annealing approach to makespan minimization on identical parallel machines. *Int. J. Adv. Manuf. Technol.* **2006**, *31*, 328–334. [CrossRef]
22. Akyol, D.E.; Bayhan, G.M. Minimizing makespan on identical parallel machines using neural networks. In *Neural Information Processing*, Proceedings of the 13th International Conference, ICONIP 2006, Hong Kong, China, 3–6 October 2006.
23. Wittrock, R.J. Scheduling parallel machines with major and minor setups. *Int. J. Flex. Manuf. Syst.* **1990**, *2*, 329–341. [CrossRef]
24. So, K.C. Some heuristics for scheduling jobs on parallel machines with setups. *Manag. Sci.* **1990**, *36*, 467–475. [CrossRef]
25. Xing, W.; Zhang, J. Parallel machine scheduling with splitting jobs. *Discret. Appl. Math.* **2000**, *103*, 259–269. [CrossRef]
26. Monma, C.L.; Potts, C.N. Analysis of heuristics for preemptive parallel machine scheduling with job setup times. *Oper. Res.* **1993**, *41*, 981–993. [CrossRef]
27. Shim, S.O.; Kim, Y.D. A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. *Comput. Oper. Res.* **2008**, *35*, 863–875. [CrossRef]
28. Kim, K.Y. Heuristics for Minimizing Makespan on Parallel Dedicated Machines with Family Setups. Master’s Thesis, Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea, 2007.
29. Yun, J.J. How do we conquer the growth limits of capitalism? Schumpeterian Dynamics of Open Innovation. *J. Open Innov. Technol. Mark. Complex.* **2015**, *1*. [CrossRef]
30. Prado, A.L.; da Costa, E.M.; Furlani, T.Z.; Yigitcanlar, T. Smartness that matters: Towards a comprehensive and human-centred characterisation of smart cities. *J. Open Innov. Technol. Mark. Complex.* **2016**, *2*. [CrossRef]
31. Yun, J.J.; Won, D.K.; Park, K.B. Dynamics from open innovation to evolutionary change. *J. Open Innov. Technol. Mark. Complex.* **2016**, *2*. [CrossRef]