IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 31, 2016, accepted September 27, 2016, date of publication October 13, 2016, date of current version March 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2616366

Real-Time Near-Optimal Scheduling With Rolling Horizon for Automatic Manufacturing Cell

CHIH-HUA HSU[1] AND HAW-CHING YANG[2]
[1]Department of Information Management, Chang Jung Christian University, Tainan City 711, Taiwan
[2]Graduate Institute of Electrical Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung 811, Taiwan

Corresponding author: C.-H. Hsu (chhsu@mail.cjcu.edu.tw)

ABSTRACT This paper presents position-based optimization methods to schedule the production of automatic cells of a wheel manufacturing factory. Real-time schedule is challenging when a cell is interrupted by various order changes. Given a sequence of orders to be scheduled, it is sorted based on an earliest due day policy, a mixed integer linear programming model is formulated, and then rolling-horizon optimization methods are used to timely find the near-optimal schedule by minimizing earliness and tardiness penalties with setup times of a manufacturing cell. In addition, an original schedule can be partial rescheduled with the preset order sequence by using the linear programming model. Experimental results show that the proposed method enables a wheel manufacturing cell to reschedule its three to five daily orders within the cycle time of a rim when there exist order changes, e.g., rush orders and customized orders. Hence, these proposed methods are promising to promptly derive the near-optimal schedule for satisfying the objective of mass customization for industry 4.0.

INDEX TERMS Earliness and tardiness cost, mixed integer linear programming, real-time scheduling, rolling horizon optimization, setup times, single machine scheduling.

I. INTRODUCTION

Through the concept of smart factory, Industry 4.0 is to achieve the goals of greater customization in mass production by facilitating more flexibility and end-to-end process automation of manufacturing systems. Mass customization is the method of "effectively postponing the task of differentiating a product for a specific customer until the latest possible point in the supply network [1]." To gain greater business agility, the manufacturing system needs to have adequate capability of dealing with increasing customization while accommodating various order changes.

In mass production, a centralized system such as MRP or EPR system is in charge of weekly or daily schedule of orders in the factory level. Scheduling determines the optimal sequence of the operations to meet the customer demands while achieves certain performance objectives [2], [3]. However, various issues, e.g., rush orders, customized orders, and machine failures, could interrupt shop floor production of the cell or line levels of a factory. There exists urgency for a cell to perform real-time scheduling and achieve the goal of rapidly responsive manufacturing.

For single machine scheduling with earliness and tardiness costs without setup times, Baker and Scudder [4] gave a general review and Garey et al. [5] proved that the optimal solution is NP-complete. The papers [6] and [7] compared different mixed integer linear programming (MILP) formulations in terms of computational efficiency with four different objective functions.

Allahverdi [8] presented a general survey on scheduling problem for different machine configurations, types of setup times, and performance indexes. In roughly 500 papers from the mid-2006 to the end of 2014 within that paper, optimization-based approach and metaheuristics are two main methodologies to solve the problems.

For one machine problems with setup times, Nogueira et al. [9] detailed six MILP formulations and evaluated their computational times with weighted completion time or tardiness costs. As the problem size increases, the computational times increases dramatically which make them unsuitable for real-time applications.

To alleviate the computational burden of solving the whole problem, Ovacik and Uzsoy [10] used rolling horizon and branch and bound algorithm to solve the problem with 100 jobs in 3 minutes of CPU time with the objective of minimizing the maximum lateness. Papers [11] and [12] considered applications in parallel machines and job shop manufacturing system.

2169-3536 © 2016 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Based on five dimensions of problem categories over 200 selected papers, Chand et al. [13] surveyed the applications of rolling horizons in operations management up to the year 2012. Imposed by the hard constraints or limits in process engineering, rolling horizon, or receding horizon control, is a popular strategy to optimize the process while maintains feasibility and please refer to the book by Kwon and Han [14] for detail applications in control engineering.

Another popular approach to solve the scheduling problem is to use metaheuristics as shown in [15] and [16] for problems with sizes up to 100 orders. For further results, please refer to [17]–[19] and the references therein.

This research considers the scheduling of an automatic cell of a wheel manufacturing factory with sequence-dependent family setup times and total weighted earliness and tardiness costs. By combining the EDD-sorted policy and rolling-horizon optimization, the proposed method could be used for timely schedule production with rush/customized orders and partial preset job sequence.

This paper is organized as follows. Section 2 describes the research problem. Different optimization strategies are formulated in Section 3. Case study are explained and compared in Section 4. Finally, Section 5 concludes the study.

II. PROBLEM DESCRIPTION

Consider the problem of daily schedule of orders for the shop floor. Each customer order of aluminum rims with different size and shape has a due date, and its order size is around 1000 to 5000 rims. In a typical shop floor of wheel manufacturing factory, there are several parallel production cells in which have similar machine configurations. For simplified illustration, Fig. 1 shows the structure of an automatic manufacturing cell.

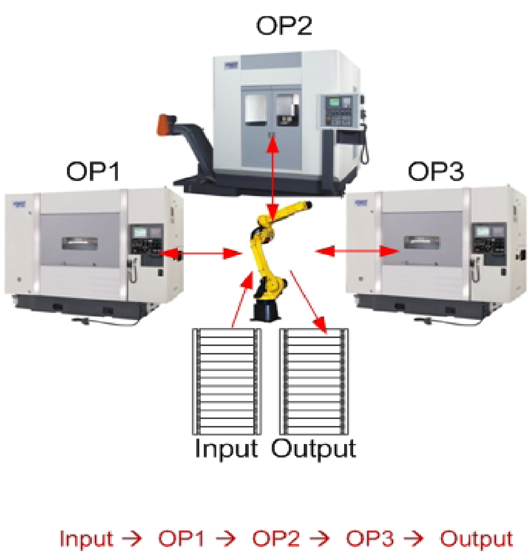

FIGURE 1. A rim is fed into the cell for manufacturing and finished after three operations.

Raw aluminum rims are fed into the system through the input end by the conveyer as shown in Fig. 1. After a rim is placed into the chamber of the vertical lathe (OP1) by the robot, the rim width and bore size of the inner side of a rim are manufactured as shown in Fig. 2 (B). The drilling machine (OP2) is to drill the five holes on the rim as shown in Fig. 2 (A). The horizontal lathe (OP3) is to trim the outer side of a rim. The rims are transferred among the machines by a robot and a new rim will be moved into the position when the previous rim is finished and transported to the next stage, i.e., there is no inventory between machines. After the operations, the rim is moved out of the chamber of OP3 by the robot and put on the conveyer at the output end which is ready for process at the next stage.

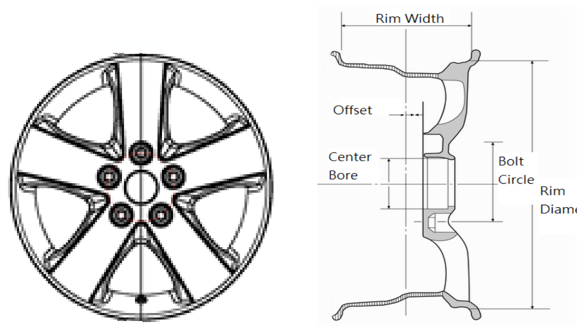

FIGURE 2. Specifications of a rim. (A) Outer side. (B) Cross section.

Before cutting, the right size of the fixtures for holding the rim firmly should be in place. If the sizes of two consecutive orders are different, the setup time for tuning and drilling machines is needed. The robot needs to use different types of fixtures for different sizes of the rims with setup time which is smaller than the machine fixture setup times. This extra setup time for the robot could be ignored because it changes simultaneously with machine fixtures.

Different size and shape of an aluminum rim needs different cutting tool for the tuning and drilling machines with setup times. Also, the tools need to be changed after cutting a certain number of rims with similar setup time. To simplify the analysis, these two setup times are combined into one setup after cutting a batch of rims. It is also assumed that each order is an integer multiple of the basic batch to make the model tractable. There are possible machine breakdowns due to ashes and cooling systems, etc., which are ignored for simpler analysis.

III. OPTIMIZATION

Given a sequence of n jobs with rim size and due days $\{d_j\}$ released at the same time, the objective is to find an optimal sequence which minimizes a certain performance index, e.g., the total weighted tardiness and earliness costs.

It is beneficial to start the processing just-in-time to minimize the cost, so the EDD policy is used as a starting point in which the sequence is sorted according to its due days and rim sizes. A similar idea was used in [11] to assign jobs for parallel machines. After the jobs are sorted, the sequence is used as an input to the following optimization module.

A. MILP

Baker and Keller [6] classified the main decision variables of an MILP into sequence position by Wagner [20], time indexing by Bowman [21], and precedence by Manne [22]. We follow closely the formulation of Wagner as in the paper by Nogueira et al. [9]. The key decision variables α_{jk} assign a set J of n jobs to a set K of n positions.

The objective function is to minimize the weighted tardiness T_j and earliness E_j with overdue penalty a_j and finished-part inventory cost b_j, respectively. As machines age, accuracy of machines might differ and its fixed purchase cost is depreciated. These effects could be captured in the coefficients of the objective functions.

$$\min_{\top} \sum_{i=1}^{n} a_j T_j + b_j E_j \tag{1}$$

subject to the constraints

$$\sum_{k \in K} \alpha_{jk} = 1, \quad \forall j \in J, \tag{2}$$

$$\sum_{j \in J} \alpha_{jk} = 1, \quad \forall k \in K, \tag{3}$$

$$\beta_{ij}^{k-1} \geq 1 - \left(2 - \alpha_{i(k-1)} - \alpha_{j(k)}\right), \quad \forall i, j \in J,\ i \neq j,\ k \in \{2, \ldots, n\}, \tag{4}$$

$$y_k \geq y_{k-1} + \sum_{j \in J} p_j \alpha_{jk} + \sum_{i \in J} \sum_{j \in J, j \neq i} \beta_{ij}^{k-1} s_{ij}, \quad \forall k \in \{2, \ldots, n\}, \tag{5}$$

$$y_1 \geq \sum_{j \in J} p_j \alpha_{j1} + \sum_{j \in J} s_{0j}, \tag{6}$$

$$C_j \geq y_k - M_k \left(1 - \alpha_{jk}\right), \quad \forall k \in K,\ j \in J, \tag{7}$$

$$T_j \geq y_k - d_j - M_k \left(1 - \alpha_{jk}\right), \quad \forall k \in K,\ j \in J, \tag{8}$$

$$E_j \geq d_j - y_k - M_k \left(1 - \alpha_{jk}\right), \quad \forall k \in K,\ j \in J, \tag{9}$$

$$\alpha_{jk} = 1, \tag{10}$$

$$C_j, T_j, E_j, y_k \geq 0, \quad \forall k \in K, j \in J, \tag{11}$$

$$\beta_{ij}^{k-1}, \alpha_{jk} \in \{0, 1\}, \quad \forall k \in K, i, j \in J,\ i \neq j. \tag{12}$$

The first two constraints denote one job at one position and one position for one job, respectively. For constraint sets (4), the binary setup variable β_{ij}^{k-1} is 1 if job i is at the position $k-1$ and job j is at the position k. The completion time y_k for position k in inequality (5) is larger than the summation of the previous completion time y_{k-1}, processing time p_j, and possible setup time s_{ij} if the rim sizes of consecutive jobs i and j are different. For the first job to be scheduled in (6), the completion time y_1 includes processing time and possible setup time s_{0j} compared with the initial configuration of the fixture. If job j is at the position k with $\alpha_{jk} = 1$, then the left-hand side of constraint sets (7) denote the completion time C_j. If job j is not at the position k with $\alpha_{jk} = 0$, a big constant M_k is needed so that these constraints are always satisfied, i.e., feasible solutions exist. The same reasoning is applied to constraint sets (8) for tardiness time T_j and constraint sets (9) for earliness time E_j of job j. As proved in [9], the number M_k is the sum of twice of the total processing times and setup times under our problem setting.

If there are rush orders coming in with certain due days, the optimization is resolved to find its new scheduling sequence. For the shop floor control, some tools need to be ready to process the order. In this case, some positions of the original optimal sequence need to be preserved, e.g., the constraint $\alpha_{jk} = 1$ for the jth job to be at the kth position in (10) could be added to the MILP to achieve this goal.

Another way to approach the rush-order problem with fixed positions is to use precedence variable x_{ji} of Baker and Keller [6], Nogueira et al. [9], and Manne [22], e.g., $x_{ji} = 1$ for all jobs i follows job j. But the difficulty is that one does not know in advance which job will show up before or after those fixed jobs. The authors [15] provided another interesting MILP by combining sequence-position and precedence variables, and it needs further investigation to understand its computational performance.

Given n jobs and time horizon h, the numbers of variables and constraints [9] of sequence-position, time-indexing, and precedence formulations are $O\left(n^3\right)$, $O\left(n^2 h\right)$, and $O\left(n^2\right)$, respectively. When the horizon is long, e.g., 1 day with horizon of 1,440 minutes, the number of decision variables of time-indexing formulation is huge.

B. EDD FIRST WITH POSSIBLE IDLE TIMES

Given a sequence of n jobs and initial configuration, it is sorted according to its due days and rim sizes and then the following optimization problem is solved with the sorted sequence.

$$\min \sum_{i=1}^{n} a_j T_j + b_j E_j \tag{13}$$

subject to constraints

$$T_j \geq C_j - d_j, \tag{14}$$

$$E_j \geq d_j - C_j, \tag{15}$$

with $C_0 = 0$. The number of variables and constraints are $O(n)$. As before, given a due day d_j and a completion time C_j of job j, the tardiness T_j and earliness E_j need to satisfy the inequalities (14) and (15), respectively.

Under this formulation, there might exist idle times between jobs to minimize the earliness costs, that is, the scheduling policy is just-in-time.

C. ROLLING HORIZON OPTIMIZATION (RHO)

As illustrated in the next section, the computational time of the optimization problem will grow dramatically as the number of jobs increases. To alleviate this problem, the full sequence of n jobs is sorted according to its due days and RHO is used to find the scheduling sequence.

The key idea of RHO is to select a subset m ($<n$) of the sorted sequence, optimize it by using an MILP

in Section III.A, implement the first position, and then repeat the process until the full sequence is completed. For the next iteration, the initial configuration and the starting time are the rim size and the completion time of the first job in the current optimized iteration, respectively. For the last iteration, the full sequence is implemented.

If the number of jobs considered in RHO is close to the full length, its total cost for all iterations will be close to the optimal cost of the whole horizon at the expense of additional computational time. This will be explained in the next simulation section to explore the tradeoff between efficiency versus accuracy.

D. NUMBER OF TARDY JOBS

When there exist difficulties to assign the coefficients or it is more important to finish the orders in time, then minimizing the number of late jobs would be a more suitable performance index.

If the objective is to minimize the number of tardy jobs, then the new problem formulation is similar to the one in Section III.A by minimizing the objective function $\sum_{j=1}^{n} U_j$, removing the constraints (8)(9), and adding the constraints $C_j \leq d_j + MU_j$, where $U_j \in \{0, 1\}, \forall j \in J$ and M is a big constant [7]. The resulting problem is still an MILP, but it has more binary variables U_js.

IV. CASE STUDY

The company currently uses the EDD policy to schedule the weekly orders with 2 rim sizes. Our simulation framework follows closely with the parametric values of its processing time, setup time, and batch size. Furthermore, randomized generated job sequences with any number of rim sizes and prefixed job sequences for rush orders could simulate more complex production environments and provide useful guidelines for future applications.

The simulation programs are written in Java to call CPLEX 12.6.3 on a Windows 7 notebook with Intel Core i3-2350M 2.3 GHz processor and 4 GB RAM.

A. SIMULATION CONDITIONS

For the company we study, the batch size is 150 rims. The operations times of the first rim of a batch at the first 2 machines and the last machine are 2 and 1.5 minutes, respectively, so the total processing time per batch is $(2 + 1.5 \times 150 =)227$ minutes. After finishing one batch, it takes about 40 minutes to change the cutting tools for the tuning and drilling machines. If the sizes of two consecutive orders are different, the setup time of the fixtures for holding the rim for tuning and drilling machines is around 5 hours. The coefficients of tardiness T_j and earliness E_j are chosen to be 10 and 1, respectively. Assume 3 different types of rim sizes with the initial configuration 1 of the fixture. Different types of rim sizes will only change the setup times s_{ij} in (5) and (6), so the decision variables and the number of constraints stay the same as 3 types of rims sizes and hence it is applicable to mass customization, that is, the maximum number of rim types could be the same as the number of orders.

The number of variables and constraints are $O(n^3)$ in an MILP, so the computational time will grow rapidly when the number of jobs is increasing. As a baseline comparison, we need to fix the number of jobs to be processed in a simulation study. According to the results in [6] and [23], the scheduling problem becomes more difficult if the jobs are likely to be tardy. Given a fixed number of jobs, 10 sequences are generated with randomized rim sizes and due days. Each sequence is then sorted by using the EDD policy and its makespan is computed. To achieve certain utilization rate of the production line, the due days are changed accordingly based on its original makespan.

B. COMPARISONS OF SCHEDULING POLICIES WITH 10 JOBS

Consider the case of 10 jobs to be scheduled. For each set of job orders, six scheduling policies are implemented including the optimal policy, EDD policy, and four rolling horizon policies with horizon m changing from 6 to 9.

In Figs. 3 and 4, the horizontal axis is the average utilization rate which is averaged over 10 simulations and the range is from 0.5 to 1.2 with an increment of 0.1. The vertical axis is the average computational time of 10 simulations in seconds. As shown in the figures, the utilization rate of the dots is not exactly an integer multiple of 0.1 which is due to rounded integer due days.

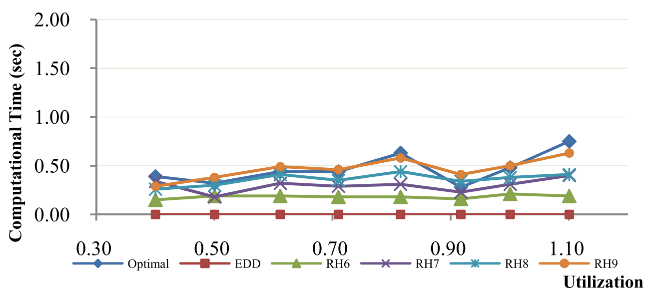

FIGURE 3. 5% lower bound of computational time with 10 jobs.

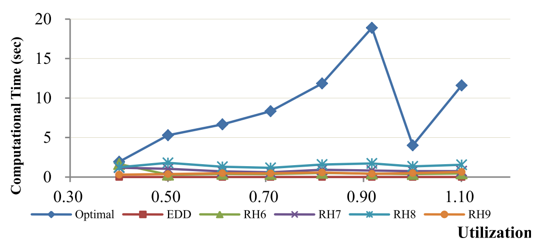

FIGURE 4. 95% upper bound of computational time with 10 jobs.

The numbers of variables and constraints of sequence-position formulations are $O(n^3)$, so the computational time in terms of job number grows nonlinearly. Given a fixed number of jobs, the optimal policy needs to solve one optimization problem for the full horizon. For RHO, it needs to solve several sub-problems for the whole horizon, so its total computational time is possibly larger than the optimal one as shown in Fig. 3.

The average costs of 10 simulations for each utilization rate and its optimality gaps as compared with the optimal policy of Wagner are drawn in Figs. 5 and 6.

FIGURE 5. Costs of different policies with 10 jobs.

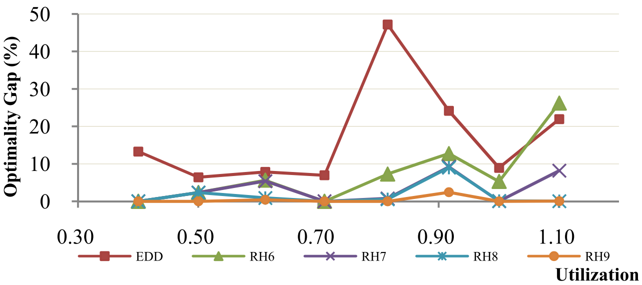

FIGURE 6. Optimality gaps of different policies with 10 jobs.

The 5% and 95% bounds of computational times in Figs. 3 and 4 are all within 20 seconds. The computational times of the EDD policy are almost zero, but its optimality gaps vary from 9% to 47%. As the horizon increases, the computational times of four rolling horizon policies increase moderately while the optimality gaps decrease from 26% to 0.05%.

C. COMPARISONS OF SCHEDULING POLICIES WITH 14 JOBS

Consider the case of 14 jobs to be scheduled. To achieve better cost performance while preserve reasonable computational times, the new rolling horizons are from 10 to 13.

The number of variables and constraints of EDD is linear, so its computation times in Figs. 7 and 8 stay almost the same with maximum time of 2 seconds while its optimality gaps vary from 14% to 161% in Table 1 which are much bigger than the range of 10 jobs. The log scale is used in Figs. 7 and 8 for easier visualization.

The number of variables and constraints of an MILP is cubic, so the maximum 95% bound of computational times of the optimal Wagner policy in Fig. 8 is 9017 seconds as compared with 20 seconds with 10 jobs.

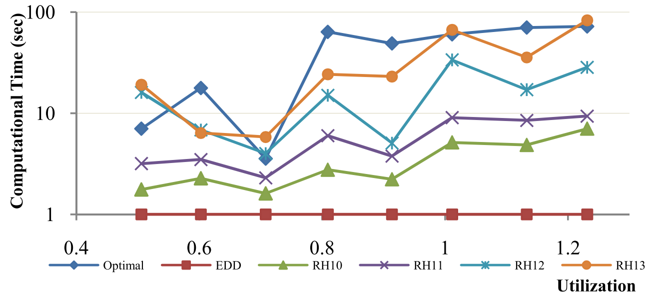

FIGURE 7. 5% lower bound of computation time with 14 jobs.

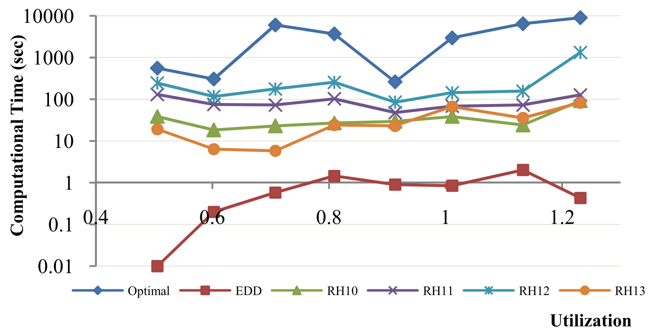

FIGURE 8. 95% upper bound of computational time with 14 jobs.

When the rolling horizon is 10, its maximum 95% bound of computational times is only 92 seconds while its worst optimality gap is 2.49% as shown in Table 1. This time is smaller than the cycle time to produce a rim which makes it suitable for real-time scheduling of orders for the shop floor when there is emergent event happening and a new schedule needs to be computed quickly.

When the horizon increases, the performance is improved as shown in Table 1 at the price of additional computational time. Hsu and Shamma [24] considered the optimization of piecewise linear objective function subject to fixed linear dynamical system and linear constraints by using manufacturing scheduling as an illustrating example. They derived sufficient condition for the receding horizon control to approach the infinite horizon objective function. Blocher and Chand [25] found out necessary and sufficient condition for the existence of forecast horizon to be optimal for the changeover scheduling problem by using cumulative productions as the fixed state variables. While in current rolling horizon optimization, the number of the dynamical equations will grow if the number of horizon increases. It is unclear how to deal with it to prove the optimality of the rolling horizon optimization. However, the simulation results indicate that choosing the horizon to be two thirds of the whole job number would result in nice performance if the due dates of jobs are sampled out of uniform distribution over an interval.

From Figs. 3 to 8, the computational times and costs are not monotone when the utilization rates increase which is probably due to 10 simulation runs. Nogueira et al. [9]

TABLE 1. Cost comparisons of six different policies with 14 jobs.

Utilization	Optimal	RH10	Opt Gap	RH11	Opt Gap	RH12	Opt Gap	RH13	Opt Gap	EDD	Opt Gap
0.5057	6303	6304	0.02%	6303	0.00%	6303	0.00%	6303	0.00%	7184	13.98%
0.6025	7754	7758	0.05%	7754	0.00%	7754	0.00%	7754	0.00%	14647	88.90%
0.7080	6621	6735	1.72%	6735	1.72%	6628	0.11%	6621	0.00%	8644	30.55%
0.8091	7310	7311	0.01%	7311	0.01%	7311	0.01%	7311	0.01%	10670	45.96%
0.9135	8564	8566	0.02%	8564	0.00%	8564	0.00%	8564	0.00%	11096	29.57%
1.0114	13228	13428	1.51%	13232	0.03%	13228	0.00%	13228	0.00%	34519	160.95%
1.1328	10888	11159	2.49%	10952	0.59%	10948	0.55%	10949	0.56%	13457	23.59%
1.2313	20120	20247	0.63%	20121	0.00%	20120	0.00%	20121	0.00%	41555	106.54%

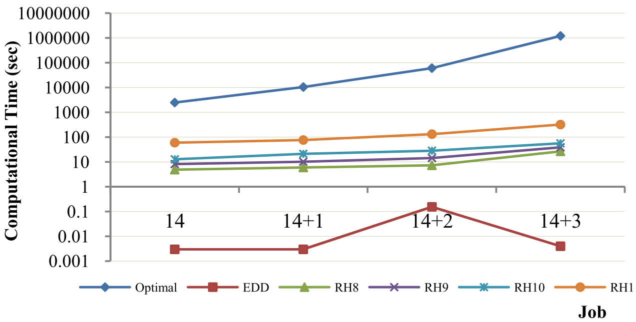

FIGURE 9. Computational time with 14 jobs and rush orders.

also considered 10 independent instances for each simulation class. For 10 jobs, 100 simulation runs are experimented and the computational times and costs are still not monotone. These results are not included to make it consistent with the presentation of the 14-job case.

D. RUSH ORDERS

Consider the case of 14 jobs with due day = {2, 2, 3, 1, 4, 4, 2, 4, 2, 2, 1, 4, 2, 4} and rim size = {1, 3, 2, 1, 1, 2, 3, 3, 2, 1, 1, 3, 3, 2}. There are three rush orders with rim size = {1, 2, 1} and due day = {3, 5, 4}. The simulation is run once for each additional rush order. With the smallest horizon of 8, its optimality gap is within 7% as in Fig. 10 while its computational times are all within 20 seconds in Fig. 9 as compared with over 100,000 seconds of the optimal policy.

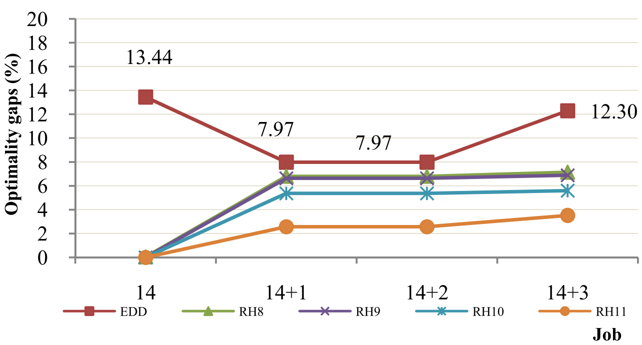

FIGURE 10. Optimality gaps of different policies with 14 jobs and rush orders.

By using an MILP for master production schedule (MPS), Wu and Chen [26] considered a numerical example to illustrate the extra cost for producing the rush order.

E. RUSH ORDER WITH PARTIAL PRESET JOB SEQUENCE

Consider the case of 11 jobs with due day = {1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4} and rim size = {1, 1, 2, 2, 1, 1, 3, 3, 3, 2, 1}. A rush order comes in with due day 1 and rim size 3. Please note that certain groups have the same due day and rim size, e.g., 2, 5 and 6, so the exchanges of their positions in a schedule do not change the value of the objective cost.

The optimal scheduling sequence of the original 11 jobs is shown in the second columns of Table 2. The due day of the seventh job is 2 which is greater than 1 of the first job, so the EDD policy is not optimal. The rim size of the seventh job is 3 which is different from the other jobs with the same due days, so it is scheduled earlier to the free time slot in the first day to avoid the setup time of the other clustering jobs.

TABLE 2. Cost comparisons of rush order with partial preset job sequence.

Job no / Seq.	11	12	12	12	12	12
	opt seq	opt seq	Fix 1	Fix 2	Fix 5	Fix 5
1	7	***12***	**7**	**7**	**7**	***12***
2	1	7	***12***	1	***12***	1
3	5	1	1	***12***	5	5
4	2	5	5	2	1	2
5	6	2	2	5	6	6
6	4	6	6	6	2	4
7	3	4	4	3	3	3
8	10	3	3	4	4	10
9	9	10	10	10	9	7
10	8	9	9	9	8	8
11	11	8	8	8	10	9
12		11	11	11	11	11
Cost	7,068	8,565	8,565	24,801	12,309	14,085

When there is a rush order with a sequence number 12, the optimal first job is the rush order as provided in the third column which changes the position of the other jobs. If some tools or preparation works are needed, this rush order would interrupt the flow of the shop floor. With our current formulations, some jobs from the original optimal sequence could be accommodated by using (10) to fix their new positions. When the first job is fixed in column 4, the cost stays the same. However, it increases dramatically from 8,565 to 24,801 when the second job needs to be fixed, too.

To demonstrate the feasibility of this approach, five jobs of even and odd numbers are fixed as shown in the last 2 columns. For rolling horizon policies, if the first job in the current iteration needs to be fixed, then it will start a new iteration with the next flexible position and keep those fixed jobs in need intact.

V. CONCLUSION

This major contribution of this study is using the EDD-sorted position-based and rolling-horizon optimization for timely schedule production of an automatic cell. The optimality gap between the optimal policy and rolling-horizon policies could be reduced within 2% if the horizon is big enough. With rush/customized orders and partial preset job sequence, the proposed method re-computes the near-optimal schedule within cycle time of a rim while preserves nice performance.

For possible future directions, one could consider random processing and setup times, machine failures, and parallel cells.

REFERENCES

[1] R. B. Chase, F. R. Jacobs, and N. J. Aquilano, *Operations Management for Competitive Advantage*, 11th ed. New York, NY, USA: McGraw-Hill, 2006.

[2] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[3] J. Y-T. Leung, Ed., *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: Chapman & Hall, 2004.

[4] K. R. Baker and G. D. Scudder, ''Sequencing with earliness and tardiness penalties: A review,'' *Oper. Res.*, vol. 38, no. 1, pp. 22–36, Feb. 1990.

[5] M. R. Garey, R. E. Tarjan, and G. T. Wilfong, ''One-processor scheduling with symmetric earliness and tardiness penalties,'' *Mathe. Oper. Res.*, vol. 13, no. 2, pp. 330–348, May 1988.

[6] K. R. Baker and B. Keller, ''Solving the single-machine sequencing problem using integer programming,'' *Comput. Ind. Eng.*, vol. 59, no. 4, pp. 730–735, Nov. 2010.

[7] A. B. Keha, K. Khowala, and J. W. Fowler, ''Mixed integer programming formulations for single machine scheduling problems,'' *Comput. Ind. Eng.*, vol. 56, no. 1, pp. 357–367, Feb. 2009.

[8] A. Allahverdi, ''The third comprehensive survey on scheduling problems with setup times/costs,'' *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 345–378, Oct. 2015.

[9] T. H. Nogueira, C. R. V. Carvalho, and M. G. Ravetti, ''Analysis of mixed integer programming formulations for single machine scheduling problems with sequence dependent setup times and release dates,'' *Quart. J. Oper. Res.*, Mar. 2014. [Online]. Available: http://www.optimization-online.org/DB_HTML/2014/07/4442.html

[10] I. M. Ovacik and R. Uzsoy, ''Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times,'' *Int. J. Prod. Res.*, vol. 32, no. 6, pp. 1243–1263, Jun. 1994.

[11] I. M. Ovacik and R. Uzsoy, ''Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times,'' *Int. J. Prod. Res.*, vol. 33, no. 11, pp. 3173–3192, Nov. 1995.

[12] M. Mohammadi and O. Poursabzi, ''A rolling horizon-based heuristic to solve a multi-level general lot sizing and scheduling problem with multiple machines (MLGLSP_MM) in job shop manufacturing system,'' *Uncertain Supply Chain Manage.*, vol. 2, no. 3, pp. 167–178, 2014.

[13] S. Chand, V. H. Hsu, and S. Sethi, ''Forecast, solution, and rolling horizons in operations management problems: A classified bibliography,'' *Manuf. Service Oper. Manage.*, vol. 4, no. 1, pp. 25–43, Jan. 2002.

[14] W. H. Kwon and S. Han, *Receding Horizon Control: Model Predictive Control for State Models*. London, U.K.: Springer-Verlag, 2005.

[15] G. Kirlik and C. Oguz, ''A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine,'' *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1506–1520, Jul. 2012.

[16] R. M'Hallah, ''Minimizing total earliness and tardiness on a permutation flow shop using VNS and MIP,'' *Comput. Ind. Eng.*, vol. 75, pp. 142–156, Sep. 2014.

[17] M. Zweben and M. S. Fox, *Intelligent Scheduling*. San Francisco, CA, USA: Morgan Kaufmann, 1994.

[18] H.-C. Chang, Y.-P. Chen, T.-K. Liu, and J.-H. Chou, ''Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-genetic algorithm,'' *IEEE Access*, vol. 3, pp. 1740–1754, 2015.

[19] M. C. V. Gallego, J. Maya, and J. R. M. Torres, ''A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan,'' *Comput. Oper. Res.*, vol. 73, pp. 132–140, Sep. 2016.

[20] H. M. Wagner, ''An integer linear-programming model for machine scheduling,'' *Naval Res. Logistics*, vol. 6, no. 2, pp. 131–140, Jun. 1959.

[21] E. H. Bowman, ''The schedule-sequencing problem,'' *Oper. Res.*, vol. 7, no. 5, pp. 621–624, Oct. 1959.

[22] A. S. Manne, ''On the job shop scheduling problem,'' *Oper. Res.*, vol. 8, no. 2, pp. 219–223, Apr. 1960.

[23] N. G. Hall and M. E. Posner, ''Generating experimental data for computational testing with machine scheduling applications,'' *Oper. Res.*, vol. 49, no. 6, pp. 854–865, Dec. 2001.

[24] C.-H. Hsu and J. S. Shamma, ''Further results on linear nonquadratic optimal control,'' *IEEE Trans. Autom. Control*, vol. 46, no. 5, pp. 732–736, May 2001.

[25] J. D. Blocher and S. Chand, ''A forward branch-and-search algorithm and forecast horizon results for the changeover scheduling problem,'' *Eur. J. Oper. Res.*, vol. 91, no. 3, pp. 456–470, Jun. 1996.

[26] M.-C. Wu and S.-Y. Chen, ''A cost model for justifying the acceptance of rush orders,'' *Int. J. Prod. Res.*, vol. 34, no. 7, pp. 1963–1974, Jul. 1996.

CHIH-HUA HSU received the B.S. degree in electrical engineering from National Sun Yat-sen University in 1989, the M.S. degree in control engineering from National Chiao Tung University, Hsinchu, in 1991, and the Ph.D. degree in aerospace engineering from The University of Texas at Austin, Austin, TX, in 1998.

He has been an Assistant Professor with the Department of Information Management, Chang Jung Christian University, Tainan City, Taiwan, since 2004. His research interests include operations research and supply chain management.

HAW-CHING YANG received the B.S. degree in engineering science from National Cheng Kung University, Tainan City, in 1989, the M.S. degree in control engineering from National Chiao Tung University, Hsinchu, in 1991, and the Ph.D. degree from the Institute of Manufacturing Engineering, National Cheng Kung University, in 2003.

He has been an Assistant Professor with the Institute of System and Control, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, since 2004, and an Associate Professor with the Graduate Institute of Electrical Engineering, National Kaohsiung First University of Science and Technology, since 2016. His research interests include production simulation, system optimization, and intelligent prognosis.