A Hybrid Method of Heuristic Algorithm and Constraint Programming for No-wait Integrated Scheduling Problem

Zhiqiang Xie[1], Xiaowei Zhang[1], Yingchun Xia[1], Jing Yang[2], Yu Xin[3]

[1] College of Computer Science and Technology, Harbin University of Science and Technology, China
[2] College of Computer Science and Technology, Harbin Engineering University, China
[3] Faculty of Electrical Engineering and Computer, Ningbo University, China
xiezhiqiang@hrbust.edu.cn, zhangxiao_526@163.com, xiayingchun@hrbust.edu.cn, yangjing@hrbeu.edu.cn
xinyhrb@qq.com

Abstract

No-wait Integrated Scheduling Problem (NISP) describes a real-life process of the non-standard products where the consideration is given to the great structure differences, processing parameter differences, no-wait constraint, and the need for further deep processing after assembly of jobs. To deal with the dynamic orders of non-standard products, the scheduling algorithm to be design should be a dynamic algorithm with the ability to deal with the above conditions. At first, the dynamic scheduling problem is transformed to a series of continuous static scheduling problem by adoption of window-based event-driven strategy, thus establishing constraint programming model targeted at minimal total tardiness and thereby proposing a hybrid method of Heuristic Algorithm and Constraint Programming (HA-CP) for the problem. In order to enhance the ability to response the dynamic orders of non-standard products, HA-CP adopts heuristic algorithm to generate a pre-scheduling solution at each dynamic event moment, so that jobs that fall into the window period are labelled as dispatched jobs, while the remaining jobs are labelled as jobs to be dispatched. To improve solution quality, the jobs to be dispatched are mapped into an operation-based constraint programming model, then, during the execution interval of dispatched jobs, constraint programming solver starts to solve the jobs to be dispatched and update the current solution if the solver gets a better solution within the execution interval. The above procedures are repeated until all jobs are scheduled. Finally, the results of simulation experiment show that the proposed algorithm is effective and feasible.

Keywords: Non-standard products, Integrated scheduling, Dynamic scheduling, No-wait, Constraint programming

1 Introduction

Production scheduling represents a very practical optimization method, through which, manufacturing can be accelerated to reduce operating consumption by reasonable arrangement of production sequence without adding any production machine [1]. With the continuous development and upgrading of related industries, competition among some enterprises has gradually shifted from fight for output and markets to supply of high-quality customized services and personalized products with demonstrating sufficient differences. In this case, the differences in structure and processing attributes between different non-standard products are obvious. With the above conditions, scheduling algorithms are more important for production workshops [2].

For different task types and machine environments, scholars have proposed abundant scheduling algorithms. Where, scheduling problem with no-wait constraints widely exists in the real-life process of steel production, computer systems, food processing, chemical industry, pharmaceutical industry, concrete products, etc. [3-4]. Many experts and scholars have studied scheduling problems with no-wait constraints. For instance, in literature [5], to solve no-wait Flow Shop scheduling problem with m-machine, a hybrid algorithm is taken based on genetic algorithm and simulated annealing. To minimize makespan of no-wait Flow Shop scheduling problem, literature [6] studied several variants of descending search and tabu search algorithm, and proposed a strategy based on a dynamic tabu list, which enhanced the algorithm's ability to jump out of local optimum to a certain extent. Literature [7] proposed a hybrid optimization algorithm based on variable neighbourhood descent and discrete particle swarm optimization to solve no-wait Flow Shop scheduling problem with two optimization goals. To minimise the weighted sum of maximum completion time and total completion time, literature [8] proposed a TOB (Trade-off Balancing) algorithm based on machine idle times. For no-wait Job Shop scheduling in which each job has its own

[*]Corresponding Author: Zhiqiang Xie; E-mail: xiezhiqiang@hrbust.edu.cn
DOI: 10.53106/160792642021092205012

processing sequence, literature [9] proposed a hybrid genetic algorithm, in which the genetic operation is treated as a subproblem and transformed into asymmetric travelling salesman problem. Literature [10] introduced artificial bee colony algorithm to solve no-wait Job Shop scheduling problem. In the above-mentioned commonly used production scheduling algorithms, no consideration is given to the great product structure differences, processing parameter differences, and the need for further deep processing after assembly of jobs in the real-life manufacturing process of non-standard products.

In fact, to quickly respond to the ever-changing market and alleviate the pressure of non-standard products in research and trial production, some enterprises have established dedicated production workshops to improve production efficiency of less-than-truckload, personalized products and non-standard products [11-12]. Where, some order-oriented SMEs organize production according to orders. During the production process, there are a large number of non-standard products that demand scribing, hand lapping, scraping and precision templates. Big differences exist in product structure and component parameters, jobs demand further deep processing after assembly, so parts cannot be predicted and prepared in advance, and production must be advanced according to BOM (Bill of Material). The problem of requiring further deep processing after jobs assembly is often referred to as Integrated Scheduling Problem (ISP) [13-15]. For ISP scheduling problem, literatures [16-17] discussed hybrid optimization method of bottleneck shifting and genetic algorithm. Literature [18] pointed out that common no-wait scheduling algorithms can only deal with the case where the number of no-wait child nodes is 1. However, in ISP, there are abundant cases in which further deep processing is required after jobs assembly, that is, the number of no-wait child nodes can be greater than 1 in ISP. Therefore, ISP with no-wait constraints is more complicated.

Some of the above-mentioned ISP scheduling algorithms do not consider the no-wait constraint, some algorithms do not consider the existence of flexible multifunctional machine in the workshop, and some do not consider dynamic scheduling needs of non-standard products and less-than-truckload custom products. Therefore, the No-wait Integrated Scheduling Problem (NISP) scheduling algorithm designed herein needs to meet the following conditions: 1) A rescheduling mechanism capable of handling dynamic orders should be designed for dynamic scheduling needs of non-standard products and less-than-truckload custom products. 2) When the number of no-wait child nodes is greater than 1, the algorithm to be designed still works. 3) The algorithm should be able to deal with the problem of flexible multifunctional machine in the workshop. 4) It should be able to deal with the problems of great differences in processing parameters and further deep processing after assembly in manufacturing process of non-standard products. To solve the above problems, this paper converts dynamic scheduling problem into a series of continuous static scheduling problem by adoption of window-based event-driven strategy, thereby proposing a hybrid optimization method of heuristic algorithm and constraint programming (HA-CP). In the end, validity of the proposed algorithm is verified by real-life instance data.

2 Problem Formulation

This paper mainly discusses No-wait Integrated Scheduling Problem (NISP) with minimal total tardiness as the goal. The NISP problem demands the following agreements: the various workbenches, machine tools, and machining centres are collectively referred to as machine; the various processing, assembly, and workflow are collectively referred to as processing; the machine environment is a dedicated trial production workshop for dynamic production; a product is composed of multiple jobs which may demand further deep processing after assembly. The product could be mapped into an operation-based task graph with tree structure constraint, as shown in Figure 1 and Table 1. In Figure 1, there are 2 products consisting of 4 jobs: Job1(v_8, v_9), Job2(v_1, v_2, v_3), Job3(v_4, v_5), Job4(v_6, v_7). Where, weight of the directed edge represents that the operation is subject to no-wait constraint. For instance, *weight*<v_6, v_7> = Φ indicates that the start time of operation v_7 cannot be earlier than the completion time of operation v_6, and there is no no-wait constraint between v_6 and v_7; *weight*<v_3, v_6> = 0 means that the start time of operation v_6 must be equal to the completion time of operation v_3, and there is a no-wait constraint between the nodes of the directed edge <v_3, v_6>.

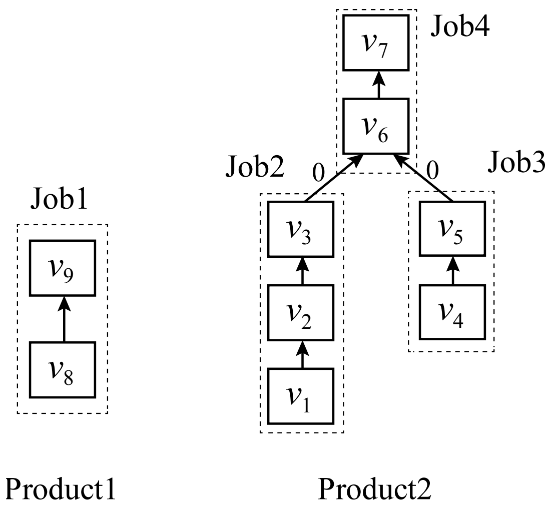


Figure 1. Instance of operation-based task graph with tree structure constraint

Table 1. Detailed data for the instance of Figure 1

Product	Job	Operation	Machine and processing time
Product1	Job1	v_8	$(d_1, 5), (d_2, 4)$
		v_9	$(d_3, 5)$
Product2	Job2	v_1	$(d_3, 4)$
		v_2	$(d_2, 6)$
		v_3	$(d_3, 3)$
	Job3	v_4	$(d_1, 8), (d_2, 5)$
		v_5	$(d_1, 9)$
	Job4	v_6	$(d_4, 4)$
		v_7	$(d_1, 3), (d_2, 4)$

Suppose $D = \{d_1, \ldots, d_m\}$ is an machine set; the product set that arrives at time t_0 is $P^0 = \{P01, \ldots, P0u\}$; $DT(P0\ i)$ represents due date of product $P0\ i(1 \le i \le u)$, $CT(P0\ i)$ represents the completion time of product $P0\ i$, then $E(P0\ i) = \max[CT(P0\ i) - DT(P0\ i), 0]$ means tardiness of product $P0\ i$; P^0 is mapped into an operation-based task set with tree structure constraint, i.e. $V^0 = \{v0\ w1, \ldots, v0\ wb\}$, and an operation $v0\ i(w_1 \le i \le w_b)$ could be processing on machine set D; $S(v0\ i, d_k, p_k) = 1$ means that operation $v0\ i$ is processed on machine $d_k(1 \le k \le m)$, which requires p_k hour. If machine d_k could not deal with operation $v0\ i$, $S(v0\ i, d_k, -) = 0$. $ST(v0\ i)$ represents the start time of $v0\ i$, $CT(v0\ i)$ represents the completion time of $v0\ i$. Therefore, the objective function for the problem studied in this paper is shown in Equation (1).

$$\min [\ \sum u\ I = 1\ E(P0\ i)\ ] \tag{1}$$

The following constraints need to be met:

If $weight\langle v0\ i,\ v0\ j\rangle = \Phi$, the operations in the directed edge should meet Equation (2).

$$ST(v0\,j) - CT(v0\ i) \ge 0 \tag{2}$$

If $weight\langle v0\ i,\ v0\ j\rangle = 0$, the operations in the directed edge should meet Equation (3).

$$ST(v0\,j) - CT(v0\ i) = 0 \tag{3}$$

Only one machine can be selected among flexible machine set for any operation $v0\ i$, which means Equation (4) should be met.

$$\sum m\ k{=}1\ S(v0\ i, d_k, -) = 1 \tag{4}$$

Time overlapping is not allowed for any two operations $v0\ i$ and $v0\ j$ $(i \ne j)$ assigned to the same machine, that is, Equation (5) should be met.

$$\begin{aligned} &nooverlap(v0\ i, v0\,j), \text{ if } S(v0\ i, d_k, -) \\ &= S(v0\,j, d_k, -) = 1 \end{aligned} \tag{5}$$

The objective functions and constraints described in equations (1)~(5) can be easily converted into a constraint programming model in Google OR-Tools or CPLEX [19-20]. Compared with common production scheduling problem, on the basis of no-wait constraints, the NISP discussed in this paper needs to further consider the great structural differences and processing parameter differences of different products, and there is also need to consider need for further deep processing after jobs assembly, so the solving difficulty is higher than common production scheduling problem [13].

3 Methodology

3.1 Dynamic Rescheduling Mechanism

As production tasks arrive in succession, to respond to these dynamic events, the scheduling algorithm needs to determine when to reschedule which jobs, then determine the processing machine for each operation and arrange the sequence of these operations on the corresponding machines. After a certain time, the finished product leaves the workshop. Repeat the above procedures until all products are finished. And, a dynamic rescheduling mechanism is exactly designed to deal with the above procedures. According to the characteristics of NISP and the actual requirements for non-standard products, an event-based rescheduling method combined with window technology is taken to handle dynamic production tasks. Figure 2 is a flow chart of the dynamic rescheduling mechanism in this paper.

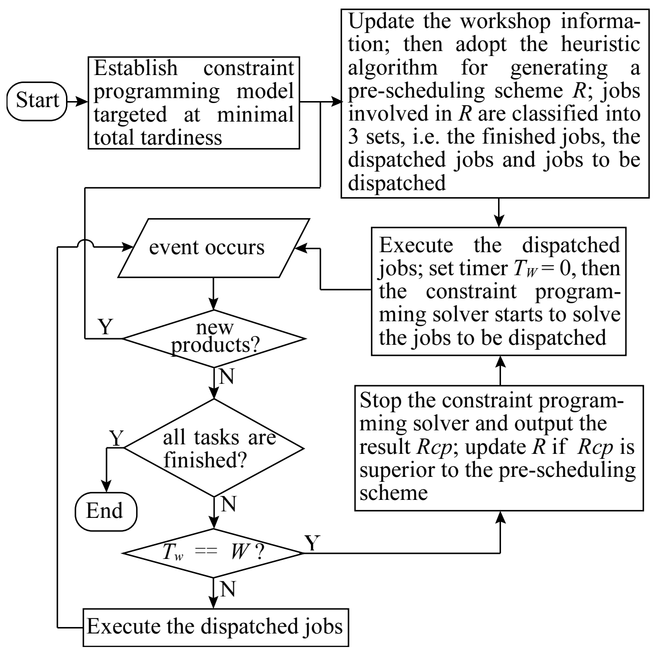

Start
Establish constraint programming model targeted at minimal total tardiness
Update the workshop information; then adopt the heuristic algorithm for generating a pre-scheduling scheme R; jobs involved in R are classified into 3 sets, i.e. the finished jobs, the dispatched jobs and jobs to be dispatched
event occurs
Execute the dispatched jobs; set timer T_W = 0, then the constraint programming solver starts to solve the jobs to be dispatched
Y
new products?
N
Stop the constraint programming solver and output the result Rcp; update R if Rcp is superior to the pre-scheduling scheme
Y
all tasks are finished?
End
N
T_w == W?
Y
N
Execute the dispatched jobs

Figure 2. Flow chart of the dynamic rescheduling mechanism

The main process is: first, define a window with a certain coverage width, for instance, $W = 2$ means a window with a width of 2 hours; at each moment of the dynamic event, the newly arrived products are mapped into a set of operations with tree structure constraint; then, update the current workshop information, and adopt the heuristic algorithm for generating a pre-scheduling scheme R; take moment of dynamic event as the starting point and W as window width, classify the jobs involved in R, i.e. the completed jobs that have finished its processing, the dispatched jobs that fall into the window period, and the rest jobs which are labelled as jobs to be dispatched; at last, during the time period when the dispatched jobs are being processed, constraint planning solver is used to reschedule the jobs to be dispatched. If the rescheduling result is superior to the pre-scheduling scheme, update the pre-scheduling scheme R.

Now, we take Figure 1 and Figure 3 as an instance for description. Product2 arrives at time t_i. At this time, Job1 is completed and all machines are idle. Product1 has only one job, so CT (Product1) = t_i, it is then labelled as completed job. For the newly arrived product Product2, a pre-scheduling scheme R is generated using heuristic algorithm. Jobs are classified based on window (t_i, $t_i + W$). In the end, Job2, Job3 are labelled as dispatched jobs, Job4 is labelled as job to be dispatched. Production task of the dispatched jobs are executed during window (t_i, $t_i + W$). Then, constraint programming solver starts to solve the jobs to be dispatched, i.e. Job4 should be rescheduled during time interval (t_i, $t_i + W$).

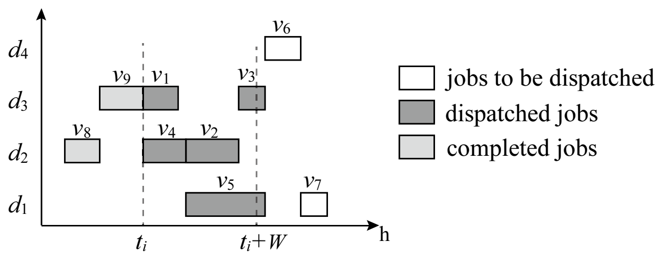


Figure 3. Instance of workshop information at time t_i

It can be known from Figure 2 that the pre-scheduling scheme R is a response of the scheduling algorithm to a dynamic event. To quickly respond to dynamic events, the scheduling algorithm in this phase imposes higher requirement for computing time. Therefore, the scheduling algorithm of this phase adopts heuristic algorithm with a faster computing speed, whose details are described in the next section. For the jobs to be dispatched, they are arranged out of the window period and can be solved using a slower but better-quality constraint programming solver. Constraint programming is an accurate solution method. The problems are pruned by algorithms such as constraint propagation to effectively reduce the search space, so that the precise algorithm has certain practicability [19-20]. Hence, the proposed HA-CP algorithm in this paper can to some extent balance the contradiction between real-time performance and solution quality in dynamic scheduling problem.

3.2 Heuristic Algorithm

In the NISP studied in this paper, a static scheduling is required at each dynamic event occurrence time. Because the number of no-wait child nodes may be greater than 1, for no-wait tasks that satisfy tree structure constraints, Equation (3) should be met. Such is a many-to-one constraint relationship. For dealing with tree structure constraints, Lei et al [13] proposed a coding method based on an operation relationship matrix table, which can guarantee that the operation sequence still satisfies the tree structure constraints after action by crossover operator and mutation operator. However, this algorithm is inapplicable if the number of no-wait child nodes is greater than or equal to 1. Therefore, on the basis of literature [13], this paper adopts grouping-based first-time fit algorithm to deal with no-wait constraint relationship. The main process is: for the operation chromosome V that satisfy the tree structure constraints and the corresponding machine chromosome M, label the operations in V as two types of no-wait operation and ordinary operation, and then divide the operation sequence V into several segments; successively schedule the operation in V; if the type of the target operation is an ordinary process, use First Fit strategy to schedule the target operation to the position where the operation is first accommodated, and label the target operation as a scheduled operation. If the type of target operation is no-wait operation, then look forwards for other operations in the same operation group, schedule them separately in the order in V, and then place the results to the previous part of the scheduling where the group of no-wait operations are first accommodated, label the group of operations as scheduled operations. Repeat the above procedures until all operations are scheduled.

Suppose there are two products shown in Figure 1 and Table 1 that need to be scheduled at the initial time. For the operation chromosome that satisfies the tree structure constraints $V = \{v_8, v_9, v_1, v_2, v_4, v_5, v_3, v_6, v_7\}$ and the corresponding machine chromosome $M = \{d_2, d_3, d_3, d_2, d_2, d_1, d_3, d_4, d_1\}$, operations in V can be labeled as three segments: ordinary operations $\{v_8, v_9, v_1, v_2, v_4\}$, no-wait operations $\{v_5, v_3, v_6\}$, ordinary operation $\{v_7\}$. Sequentially schedule the operations in V, v_8 is processed on d_2, the gap where v_8 is first accommodated is [0, 4], then schedule v_8 to this position, as shown in Figure 4(a); Completely scheduled $\{v_8, v_9, v_1, v_2, v_4\}$ is shown in Figure 4(b); $\{v_5, v_3, v_6\}$ is a group of no-wait operations, and its individual scheduling results must be closely linked together. Then, find the position from the gap in Figure 4(b) where the group of no-wait operations is first accommodated, the result is shown in Figure 4(c); the final scheduling result is shown in Figure 4(d). In this way, the conversion from V, M to a scheduling solution is completed.

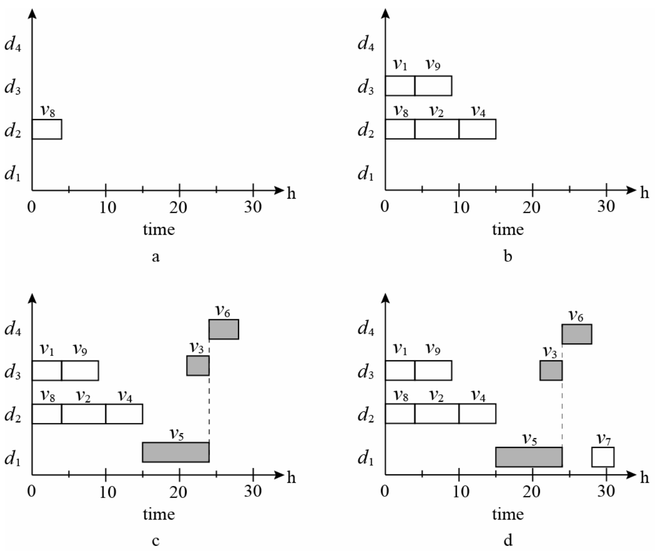


Figure 4. Instance of grouping-based first-time fit algorithm

Literature [13] is a genetic algorithm, and the chromosome is an operation sequence V and corresponding machine sequence M that satisfy the tree structure constraints. The genetic algorithm is sensitive to the initial population and causes important influence on the scheduling results. Topcuoglu et al [21] proposed the EFT (Earliest Finish Time) scheduling strategy, which is a simple and fast task scheduling algorithm. EFT strategy sorts the operations based on upward path value, and the value is calculated by average processing time of each operation. This paper proposes a method for generating random initial population. For process v_i, it supports flexible processing on multiple machines. The processing time can be consistent or different. Suppose the upper and lower limits of the processing time are $[p_{min}, p_{max}]$, then the random value in this interval can be taken as a reference value for the processing time in process v_i. Then, randomly assign a random processing time reference value belonging to respective interval for all processes, calculate the process scheduling order at this time, and schedule according to EFT strategy. At this time, a pair of sequences can be obtained, i.e. the operation sequence V and the corresponding machine sequence M that satisfy the tree constraint relationship. Repeat this process until the quantity requirements are met.

3.3 Constraint Programming Solver

Google OR-Tools is an open source software suite for optimization, tuned for tackling the world's toughest problems in vehicle routing, flows, integer and linear programming, and constraint programming [19]. We use Google OR-Tools CP-SAT solver as the constraint programming solver in this work.

4 Simulation Experiment

The HA-CP algorithm designed in this paper adopts a hybrid optimization method combining heuristic algorithm and constraint programming, which has strong adaptability to less-than-truckload personalized products with great structural differences and obvious differences in processing parameters. In order to verify HA-CP performance, Workflow Generator [22] is used as a reference to write an Instance Generator for NISP problems in this work. The main configuration of the experimental computer is: Dell PowerEdge R720 (Intel Xeon E5-2660 v2@2.6GHz * 2).

We utilize the following methods, ORMT (Integrated Scheduling Algorithm based on an Operation Relationship Matrix Table) [13], VCLDC (Product Comprehensive Scheduling Algorithm based on Virtual Component Level Division Coding) [23] as the control methods. As ORMT and VCLDC do not consider the no-wait constraints and the dynamic scheduling demand, we give the following improvement for the methods.

Improved ORMT. Firstly, the dynamic scheduling problem is transformed to a series of continuous static scheduling problem by adoption of window-based event-driven strategy. Then, for a particular static scheduling problem, to convert operation chromosome V, machine chromosome M to a scheduling solution that meet no-wait constraint, it adopts grouping-based first-time fit algorithm (in **3.2**) to generate the solution. At each dynamic event moment, reschedule all target jobs according to the above procedures.

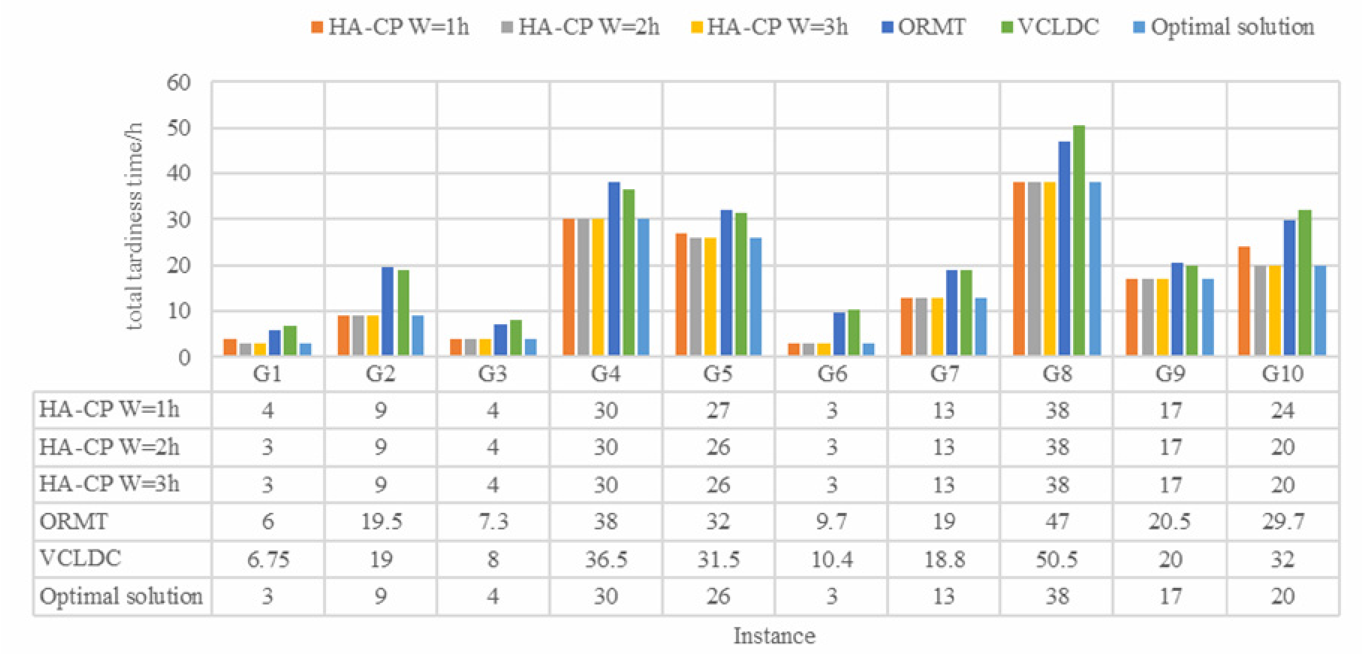
Improved VCLDC. Same as Improved ISA-ORMT.

To verify the performance of HA-CP, each instance is simulated 10 times using HA-CP, ORMT and VCLDC respectively. From the one aspect of average of the 10 running results, compared with the control methods.

Experiment 1: HA-CP adopts a heuristic algorithm for scheduling at event-driven moment, and uses a constraint programming solver to optimize the dispatched jobs during the time period when the dispatched jobs are processed. Where, set the parameters of the heuristic algorithm of HA-CP according to literature [13], i.e. population size is 100, maximum generation count is 200, probability of crossover operator is 0.6 and probability of mutation operator is 0.01.

In Figure 2, the window period W when the dispatched jobs are processed is used to solve the jobs to be dispatched. It is easy to know that for a longer window period W, the number of dispatched jobs in the pre-scheduling scheme is increased, and the number of remaining jobs to be dispatched is decreased relatively. At this time, the problem size decreases and the running time for the constraint programming solver turns longer, thus the probability that the constraint programming solver gets the optimal solution increases.

However, the probability of occurrence of dynamic events will increase in an excessively long window period, and frequent occurrence of dynamic events will interrupt the constraint programming solver. Therefore, the value of window period *W* should fully consider the contradiction between real-time performance and solution quality in dynamic scheduling. To check *W* value and the solution quality of the proposed HA-CP, G1-G10 have been generated by the Instance Generator. G1-G10 are characterized by the following parameters: device count *m* = 10, product count of each instance is *u* = 5, average operation count per product is *w* = 50, average no-wait constraint count per product is 10. There are 5 products in each instance, and the arrival hour and due hour of each instance are {0 0 24 24 48} and {50 70 80 80 120} respectively. HA-CP algorithm takes *W* = {1, 2, 3} parameters to simulate the scheduling G1-G10 respectively. Performance indicators of the algorithm is verified under different parameters, and the results are shown in Figure 5.



	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
HA-CP W=1h	4	9	4	30	27	3	13	38	17	24
HA-CP W=2h	3	9	4	30	26	3	13	38	17	20
HA-CP W=3h	3	9	4	30	26	3	13	38	17	20
ORMT	6	19.5	7.3	38	32	9.7	19	47	20.5	29.7
VCLDC	6.75	19	8	36.5	31.5	10.4	18.8	50.5	20	32
Optimal solution	3	9	4	30	26	3	13	38	17	20

Figure 5. The results of G1-G10 generated by HA-CP with different *W* value

Figure 5 shows the simulation results of G1-G10. It can be seen that HA-CP have reached the optimal solutions in 7 instances when *W* = 1, and HA-CP have reached the optimal solutions in all instances when *W* = 2 or 3.

Experiment 2: A workshop of a Shanghai power equipment company is mainly engaged in the R&D and production of complex mechanical and electrical products. Its workshop adopts less-than-truckload, customized processing. In the dynamic production process, the production task of processing new orders is normal. In this paper, HA-CP algorithm takes *W* = {1, 2, 3} respectively to simulate and schedule production tasks in a certain time period of the workshop. The production tasks are characterized by the following parameters: count of device *m* = 25, total product count *u* = 18, total operation count is 658, total count of no-wait constraint is 133. There are 18 products in experiment 2, and the arrival hour and due hour are {0 0 0 0 0 0 0 10 15 15 25 25 35 45 50 55 65 65} and {40 40 60 50 46 65 65 59 55 60 55 60 40 60 58 50 53 52} respectively. Figure 6 is the result of experiment 2.

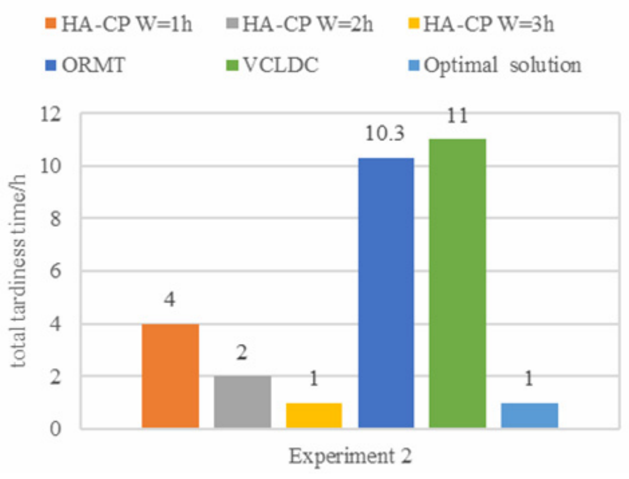


Figure 6. The result of experiment 2

Compared with the control methods, the HA-CP gets better results in all instances of the 2 simulation experiments, because HA-CP makes full use of the time period when the scheduled jobs of the pre-scheduled solution are executing and optimizes the jobs to be scheduled with constraint programming solver during the time period. Therefore, the proposed HA-CP in this work is effective and feasible.

5 Conclusion

The differences between NISP scheduling and commonly used scheduling are described. A constraint programming model is descripted for NISP problem. A hybrid optimization method combining heuristic algorithm and constraint programming is put forward. The conclusions are drawn as follows: (1) the proposed HA-CP optimization method provides an effective and feasible technical solution; (2) solving the jobs outside the window W using constraint programming solver within the processing period of dispatched jobs inside the window W could balance to a certain extent the contradiction between real-time performance and solution quality in dynamic scheduling; (3) seen from the result of experiment 2, HA-CP could improve the solution quality by adjusting the window duration value W.

In fact, constraint programming solver is accurate method, and its time complexity increases exponentially with the problem scale. In order to alleviate the running time of large-scale instance, multi-level window technology can be adopted. That is, the pre-scheduling scheme could be subdivided for multiple levels of windows to reduce the number of jobs in a window. By making full use of the time period in which the tasks of the previous window are being processed, the tasks of the next window are solved by using constraint programming solver. Therefore, this work can serve as the research basis for NISP dynamic scheduling problem.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No.: 61772160, 61602133), the China Postdoctoral Science Foundation (Grant No.: 2016M591541), the Research Fund for the Doctoral Program of Higher Education of China (Grant No.: 20122304110012), the Scientific and Technological Research Project of Heilongjiang Provincial Education Department (Grant No.: 12531105), the Postdoctoral Scientific Research Start Fund Project in Heilongjiang Province (Grant No.: LBH-Q13092), the Postdoctoral Science Foundation of Heilongjiang Province (Grant No.: LBH-Z15096).

References

[1] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Book Reviews, *IIE Transactions*, Vol. 28, No. 8, pp. 695–697, September, 2016.

[2] X. Zhang, Z. Xie, Y. Xin, J. Yang, Integrated Scheduling Algorithm of Two Workshops based on Optimal Time, *Computer Integrated Manufacturing Systems*, Vol. 23, No. 9, pp. 1938–1949, September, 2017.

[3] A. Allahverdi, A Survey of Scheduling Problems with No-wait in Process, *European Journal of Operational Research*, Vol. 255, No. 3, pp. 665–686, December, 2016.

[4] C. J. Schuster, J. M. Framinan, Approximative Procedures for No-Wait Job Shop Scheduling, *Operations Research Letters*, Vol. 31, No. 4, pp. 308–318, July, 2003.

[5] T. Aldowaisan, A. Allahverdi, New Heuristics for No-Wait Flowshops to Minimize Makespan, *Computers and Operations Research*, Vol. 30, No. 8, pp. 1219–1231, July, 2003.

[6] J. Grabowski, J. Pempera, Some Local Search Algorithms for No-Wait Flow-Shop Problem with Makespan Criterion, *Computers & Operations Research*, Vol. 32, No. 8, pp. 2197–2212, August, 2005.

[7] Q.-K. Pan, M. F. Tasgetiren, Y.-C. Liang, A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem, *Computers & Operations Research*, Vol. 35, No. 9, pp. 2807–2839, September, 2008.

[8] H. Ye, W. Li, B. R. Nault, Trade-off Balancing Between Maximum and Total Completion Times for No-Wait Flow Shop Production, *International Journal of Production Research*, Vol. 58, No. 11, pp. 3235–3251, June, 2020.

[9] J. C. H. Pan, H. C. Huang, A Hybrid Genetic Algorithm for No-Wait Job Shop Scheduling Problems, *Expert Systems with Applications*, Vol. 36, No. 3 PART 2, pp. 5800–5806, April, 2009.

[10] S. Sundar, P. N. Suganthan, C. T. Jin, C. T. Xiang, C. C. Soon, A Hybrid Artificial Bee Colony Algorithm for The Job-Shop Scheduling Problem with No-Wait Constraint, *Soft Computing*, Vol. 21, No. 5, pp. 1193–1202, March, 2017.

[11] B. Wen, Z. Zhou, Q. Han, B. Su, Important Role of Products Design of Modern Machinery in the Research and Development of New Products: Study on the Threefold Synthesis Method "Dynamic Design, Intelligent Design and Partial Virtual Design" Face to Products Generalization Quality, *Chinese Journal of Mechanical Engineering*, Vol. 39, No. 10, pp. 43–52, October, 2003.

[12] J. H. Cui, J. Chen, Y. F. Deng, Q. Liu, L. Sun, Research for New Product Trial Production Task Scheduling Method in Machinery Manufacturing Enterprises, *Journal of Jiangnan University* (*Natural Science Edition*), Vol. 14, No. 3, pp. 326–332, June, 2015.

[13] Q. Lei, W. Guo, Y. Song, Integrated Scheduling Algorithm based on an Operation Relationship Matrix Table for Tree-Structured Products, *International Journal of Production Research*, Vol. 56, No. 16, pp. 5437–5456, February, 2018.

[14] Z. Xie, S. Hao, G. Ye, G. Tan, A New Algorithm for Complex Product Flexible Scheduling with Constraint Between Jobs, *Computers & Industrial Engineering*, Vol. 57, No. 3, pp. 766–772, October, 2009.

[15] Z. Q. Xie, X. H. Zhang, Y. Xin, J. Yang, Time-selective Integrated Scheduling Algorithm Considering Posterior Processes, *Zidonghua Xuebao/Acta Automatica Sinica*, Vol. 44, No. 2, pp. 344–362, February, 2018.

[16] P. Ivens, M. Lambrecht, Extending the Shifting Bottleneck Procedure to Real-Life Applications, *European Journal of*

Operational Research, Vol. 90, No. 2, pp. 252–268, April, 1996.

[17] F. Shi, S. Zhao, Y. Meng, Hybrid Algorithm based on Improved Extended Shifting Bottleneck Procedure and GA for Assembly Job Shop Scheduling Problem, *International Journal of Production Research*, Vol. 58, No. 9, pp. 2604–2625, May, 2020.

[18] Z. Xie, Y. Xin, J. Yang, No-Wait Integrated Scheduling Algorithm based on Reversed Order Signal-Driven, *Journal of Computer Research & Development*, Vol. 50, No. 8, pp. 1710–1721, August, 2013.

[19] L. Perron, V. Furnon, *OR-Tools*, Google LLC, https://developers.google.com/optimization, Accessed July, 2019.

[20] IBM, *the CPLEX User's Manual 12.8.0*, Compagnie IBM France, https://www.ibm.com, Accessed December, 2017.

[21] H. Topcuoglu, S. Hariri, M. Y. Wu, Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3, pp. 260–274, March, 2002.

[22] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and Profiling Scientific Workflows, *Future Generation Computer Systems*, Vol. 29, No. 3, pp. 682–692, March, 2013.

[23] S. K. Zhao, Q. Han, G. C. Wang, Product Comprehensive Scheduling Algorithm based on Virtual Component Level Division Coding, *Computer Integrated Manufacturing Systems*, Vol. 21, No. 9, pp. 2435–2445, September, 2015.

Biographies

Zhiqiang Xie obtained his M.S. and Ph.D. at Harbin University of Science and Technology in July 2003 and 2009 respectively. Since 2015, he works as a Ph.D. supervisor for Harbin University of Science and Technology, China. He is an outstanding member of China Computer Federation. His main research field is integrated scheduling algorithm. Until now, he has published more than 100 papers.

Xiaowei Zhang, Ph.D. student. Currently he is in Harbin University of Science and Technology, China. He received his B.S. and M.S. at Harbin University of Science and Technology in July 2005 and 2012 respectively. His main research interests include cloud scheduling algorithm and integrated scheduling algorithm.

Yingchun Xia, Ph.D. student. He received the M.S. Degree from Harbin University of Science and Technology in China. His main research interest is integrated scheduling algorithm.

Jing Yang is a professor and PhD supervisor at Harbin Engineering University in China, she graduated from Northeast Heavy Machinery Institute in July 1984 and received her PhD in computer application technology from Harbin Engineering University in June 2006. Her research covers big data analysis and privacy protection, social networks, information security, and scheduling algorithms.

Yu Xin, Ph.D., a professor at Ningbo University, China. He is a member of China Computer Federation. His main research interests include database, data mining, and privacy preservation.