applied sciences

MDPI

Article

A Genetic Regulatory Network-Based Method for Dynamic Hybrid Flow Shop Scheduling with Uncertain Processing Times

Youlong Lv, Jie Zhang * and Wei Qin

School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; xsygll@sjtu.edu.cn (Y.L.); wqin@sjtu.edu.cn (W.Q.)

* Correspondence: zhangjie@sjtu.edu.cn; Tel.: +86-27-3420-6292

Academic Editors: Naiqi Wu, Mengchu Zhou, Zhiwu Li and Yisheng Huang

Received: 2 November 2016; Accepted: 19 December 2016; Published: 4 January 2017

Abstract: The hybrid flow shop is a typical discrete manufacturing system. A novel method is proposed to solve the shop scheduling problem featured with uncertain processing times. The rolling horizon strategy is adopted to evaluate the difference between a predictive plan and the actual production process in terms of job delivery time. The genetic regulatory network-based rescheduling algorithm revises the remaining plan if the difference is beyond a specific tolerance. In this algorithm, decision variables within the rolling horizon are represented by genes in the network. The constraints and certain rescheduling rules are described by regulation equations between genes. The rescheduling solutions are generated from expression procedures of gene states, in which the regulation equations convert some genes to the expressed state and determine decision variable values according to gene states. Based on above representations, the objective of minimizing makespan is realized by optimizing regulatory parameters in regulation equations. The effectiveness of this network-based method over other ones is demonstrated through a series of benchmark tests and an application case collected from a printed circuit board assembly shop.

Keywords: hybrid flow shop; uncertain processing time; genetic regulatory network; event-driven rescheduling strategy

1. Introduction

The Hybrid Flow Shop (HFS) is a typical discrete manufacturing system in which a set of jobs passes through a series of production stages to complete required operations. The stages are composed of parallel machines to protect the job flow from being blocked by a single machine [1–3]. The operation processing time varies with different machines because capacities of parallel machines are normally unrelated at each stage [4,5]. This type of workshop exists in various industries, which include Printed Circuit Board (PCB) assembly, textile production and automobile assembly [6–9].

Since the first HFS problem was described by Salvador [10], vast amounts of academic work have been carried out on HFS scheduling due to its complexity and practical relevance [11–15]. Different kinds of methods (e.g., exact methods, heuristics and metaheuristics) were proposed to minimize a variety of objectives, which include the maximum completion time, the maximum flow time, the number of late jobs [16–19]. In terms of minimizing the maximum completion time, i.e., makespan, Mirsanei et al. [20] proposed a simulated annealing algorithm to solve the HFS scheduling problem featured with sequence-dependent setup times. Wang et al. [21] developed an efficient dispatching rule together with several local search heuristics in a semiconductor manufacturing system. Wang et al. [22] proposed a branch-and-bound algorithm to investigate the two-stage HFS scheduling problem in a no-wait environment. Komaki et al. [23] developed several algorithms featured with a new lower

bound to minimize the makespan in a specific HFS consisting of two machining stages and one assembly stage.

In most of these approaches, the processing times were assumed to be deterministic in HFS scheduling. However, in a real workshop, the differences between machines, worker skill levels and material qualities make the processing times constantly changing [24]. These variations might lead to increased objective values during the execution of predetermined plans, which occurs frequently in practical production [25,26]. To deal with this situation, dynamic scheduling used for HFS scheduling problems consists of three major ways: proactive, reactive and predictive-reactive [27,28].

The proactive scheduling strategy focuses on generating a robust solution that satisfies performance requirements predictably [29]. This strategy normally uses fuzzy theory or statistical analysis to construct the probability distribution of processing times and applies these distributions to HFS scheduling [30,31]. Nevertheless, it is difficult to ensure the predicted performance in practical situations. In a reactive scheduling strategy, all the decisions are made in real time to react rapidly to dynamic events of processing time uncertainty, e.g., by using dispatching rules and game theory approaches [32–34]. However, this strategy can hardly ensure the global performance of HFS scheduling because of the limits placed on computational burden in real-time scheduling. Predictive-reactive scheduling generates a predictive plan based on an expected production environment and revises the plan if dynamic events cause the actual situation differs from expectation [35,36]. This strategy has been widely used because it can achieve a good balance between global performance of solutions and in-time reaction to unexpected situations.

In a predictive-reactive scheduling, predetermined plans are revised by rescheduling in a rolling horizon, rather than in the whole horizon, in order to avoid too much computational efforts [37]. Three types of rolling horizon rescheduling mechanisms, including cycle-driven rescheduling, event-driven rescheduling and mixed driven rescheduling, are developed as an alternative. Of these mechanisms, the event-driven one is most widely used because it can respond to event-related disturbances in a real time. In this mechanism, the reactive scheduling is normally motivated by disturbances beyond a specific level in order to reduce the rescheduling for frequently occurring events.

However, it is currently challenging to find out a method that can achieve a good balance between solution quality as well as computational time when it is necessary to consider rescheduling in HFSs. Heuristic methods obtain a feasible solution quickly, but can hardly ensure the solution quality because they fail to take all the aspects of HFS scheduling into account. Alternatively, metaheuristic methods are capable of finding out optimal solutions owing to their general procedures. Nevertheless, these procedures require too much computational effort and will make the rescheduling plan not a real-time one. Consequently, it is necessary to develop an efficient rescheduling method for the dynamic scheduling of HFS.

The Genetic Regulatory Network (GRN) is a structured network that describes the regulation of gene expression in cells [38]. It has been attracting increased attention because of its excellent description capacity, and various methods, including directed graphs, Boolean networks, rule-based formalisms, qualitative differential equations, Bayesian networks and partial differential equations are currently alternative to construct such a network [39]. A GRN has at least the following three elements in common: genes, gene regulations and gene expression procedures [40]. Each gene has two alternative states (i.e., the expressed state and the unexpressed state). If a gene is in the expressed state, it has inhibitory effects on the states of other ones, which is the primary form of gene regulations. Based upon these regulations, the gene expression procedure converts iteratively certain genes in the unexpressed state into ones in the expressed state if there are few inhibitory effects on these genes. According to genes in the expressed state, the GRN finally determines a specific morphology for the related cell. For instance, Figure 1 illustrates a GRN composed of three genes and constructed by differential equations. In this network, μ_z (z = 1–3) represents the state of gene z; σ_z (z = 1–3) is the inhibition coefficient that describes quantitatively the inhibitory effects on gene z; ε_1, ε_2 and ε_3 are regulatory parameters in gene regulation equations. At the beginning, all genes are in the unexpressed

state. Based on the inhibition coefficients determined by regulatory parameters, the gene expression procedure converts certain genes to the expressed state at discrete moments, and finally ends with the gene states (0,1,1).

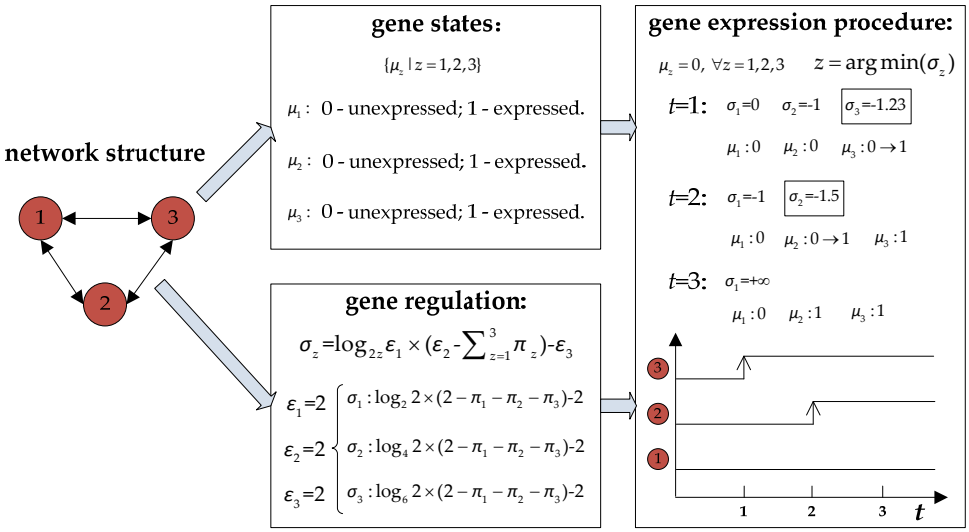

Figure 1. An example of GRN (Genetic Regulatory Network) composed of three genes.

In a HFS scheduling problem, the binary decision variable is similar to the state of a gene. The decision variable interaction in constraints and objectives has an analogous form with gene regulations. Therefore, genes in a GRN are used to express decision variables, and gene regulations are adopted to describe constraints and objectives. In this way, a gene expression procedure depending on the regulations appropriately can obtain a feasible solution, in which the state of a gene indicates the assigned value of a related decision variable. In general, the differential equation method is most suitable for such a GRN because it provides the detailed description of gene regulations in a quantitative way [41]. In such an equation, undetermined regulatory parameters can be further optimized to obtain a near-optimal solution within the reasonable computational time because the number of these parameters is limited. Based on the mapping relationship illustrated in Figure 2, a GRN-based method is thus proposed for HFS rescheduling.

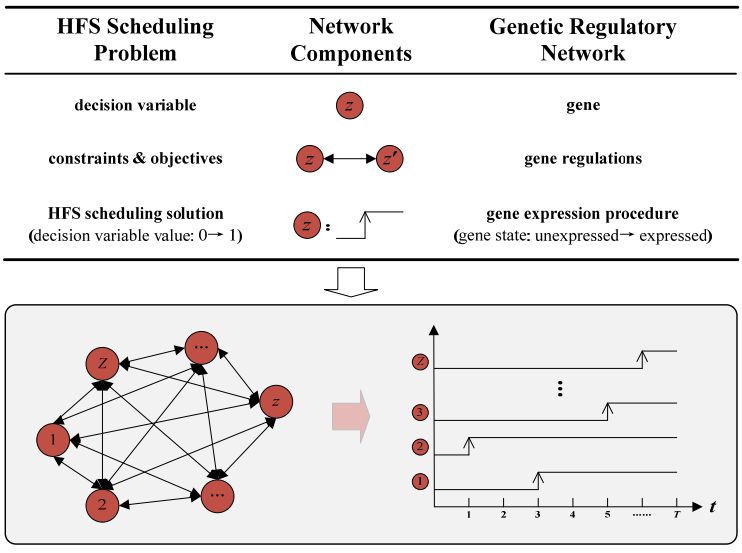

Figure 2. Mapping between the HFS (Hybrid Flow Shop) scheduling problem and a GRN.

In general, the predictive-reactive scheduling strategy based on an event-driven rescheduling mechanism is applicable for HFS scheduling with uncertain processing times. However, it is difficult to obtain a high-quality solution in real time when reactive scheduling is required. This paper thus proposes a GRN-based method to solve this dynamic scheduling problem and uses computational experiments to validate this method. The rest of this paper is organized as follows. Section 2 presents the HFS scheduling problem with uncertain processing times. Section 3 introduces the event-driven rescheduling strategy. In Section 4, the GRN-based rescheduling method is presented. Section 5 gives computational experiments and discussions. The conclusion is outlined in Section 6.

2. Hybrid Flow Shop Dynamic Scheduling Problem

The HFS scheduling problem is to determine the sequence of jobs entering the first stage and assign jobs to alternative machines at every stage. Its objective is to minimize the makespan of jobs in order to increase machine utilization and guarantee on-time job delivery simultaneously. The following assumptions are taken into consideration in this problem:

1. Each job and each machine are available at the initial time.
2. Each job passes through multiple production stages to complete operations.
3. One or more parallel machines are available at each stage.
4. Parallel machines require different processing times for the same operation.
5. Each machine is not able to process more than one job at the same time, and cannot be interrupted until the operation on this job has been completed.
6. A job can enter the next stage if its operation at current stage has been accomplished.
7. Processing times are uncertain, and their actual values may be different from the expected ones.
8. A machine requires changeover time if it needs to process two jobs of different types consecutively.
9. Operations of a job have no effect on those of other jobs.

Table 1 lists the notations used in this problem. In this table, t_{mki} represents the operation processing time, which is uncertain in an actual environment. Erlang distribution is a common way to construct the distribution of processing times based on the queuing theory. Based on this distribution, the possibility density function of a processing time t_{mki} is as follows:

$$f_{tmi}(t) = (v\lambda)\frac{(v\lambda t)^{v-1}}{(v-1)!}e^{-v\lambda t},\ v \in N \tag{1}$$

where t represents the actual processing time, $f_{tmi}(t)$ represents the v-order Erlang distribution of t_{mki}, $E(t) = 1/\lambda$. y_{rm} and x_{rki} are decision variables representing the sequence of jobs entering the first stage and the processing equipment of each job at every stage. The dynamic scheduling should find appropriate decision variable values to minimize the makespan when the processing times are constantly changing throughout the HFS production.

Table 1. Problem's notations.

Notations	Definitions
Sets	
$\{1,\cdots,m,\cdots,M\}$	Set of job types
$\{1,\cdots,i,\cdots,I\}$	Set of production stages
$\{1,\cdots,k,\cdots,K_i\}$	Set of parallel machines at stage i
$\{1,\cdots,r,\cdots,R\}$	Set of waiting processing jobs
Parameters	
t_{mki}	Operation processing time of job type m on the kth machine at stage i
$c_{mm'ki}$	Changeover time required by the kth machine at stage i to operate on job type m after job type m'
d_m	Production volume of job type m, $R = \sum_{m=1}^{M} d_m$

Table 1. *Cont.*

Notations	Definitions
Variables	
y_{rm}	Binary variable: 1, if the rth job entering the first production stage belongs to job type m; 0, otherwise
x_{rki}	Binary variable: 1, if the rth job entering the first production stage is processed on the kth machine at stage i; 0, otherwise
a_{rki}	The time instant the kth machine of production stage i to be available for the rth job
b_{rki}	The job type processed by the kth machine of production stage i before a_{rki}

3. Event-Driven Rescheduling Strategy

The predictive-reactive strategy is adopted to realize the dynamic scheduling process in a HFS. According to this strategy, a predictive plan is first developed based on the expected value of processing times. However, there will be a difference between the actual situation and the predictive plan in terms of operation processing times. The event-driven rescheduling mechanism evaluates the disturbance caused by these events and judges the necessity of a reactive scheduling. As shown in Figure 3, taking the objective of minimizing makespan into consideration, the judgment is based on whether delivery time deviation of jobs exceeds a specific tolerance. Delivery time deviation of jobs is calculated as follows:

$$\delta_{nm} = |rC_{nmI} - C_{nmI}|/C_{nmI}, \tag{2}$$

where C_{nmI} and rC_{nmI} represent expected and actual delivery time of the nth batch of job type m, respectively. If the deviation satisfies $\delta_{nmI} > \delta_{\max}$, a rescheduling is required, otherwise, the predictive plan is kept. A large $\delta_{\max}$ causes the reactive scheduling insensitive to dynamic events, whereas a small $\delta_{\max}$ leads to frequent rescheduling. Thereupon, delivery deviation tolerance $\delta_{\max}$ is an important parameter in the rescheduling strategy.

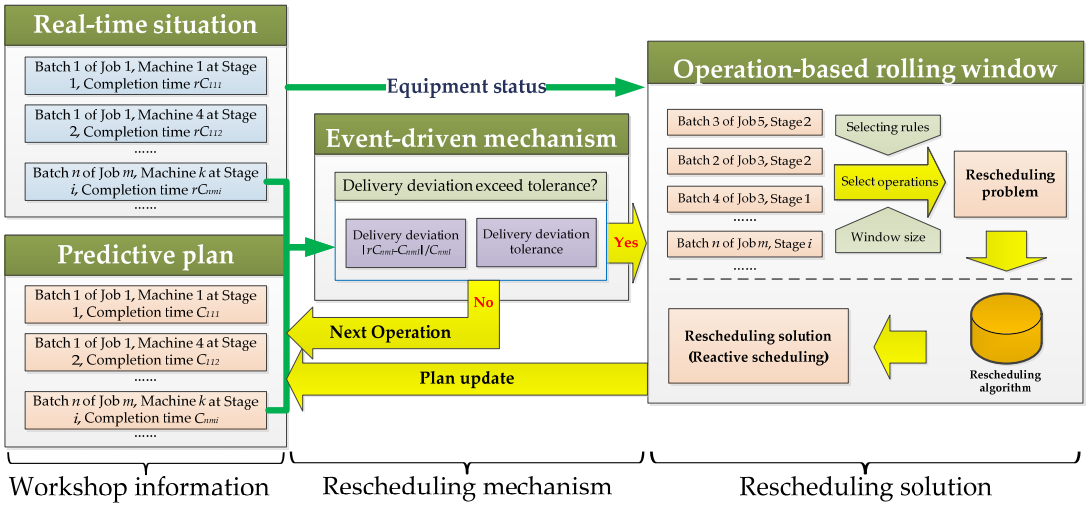

Figure 3. Event-driven rescheduling strategy.

Moreover, the reactive scheduling is performed within an operation-based rolling window because the processing times are related to operations. As illustrated in Figure 4, the window keeps removing the completed operations and adding the waiting processing ones based on real-time shop information. $S(l)$ represents the set of p operations within the lth window. $CS(l)$ represents the set of completed operations. $ES(l)$ is the set of waiting processing operations. The rolling window forms the rescheduling problem based on $S(l)$ once the reactive scheduling is regarded as necessary. The rescheduling algorithm solves this problem without interrupting the ongoing operations and then generates a new predictive plan based on $CS(l)$ and $ES(l)$, as shown in Figure 3. For each window, a larger p makes the rescheduling

problem more complex, but enables better global optimization. On the contrary, a smaller p decreases the computational effort by compromising solution quality. Consequently, the number of operations within each window is also a key parameter in the rescheduling strategy.

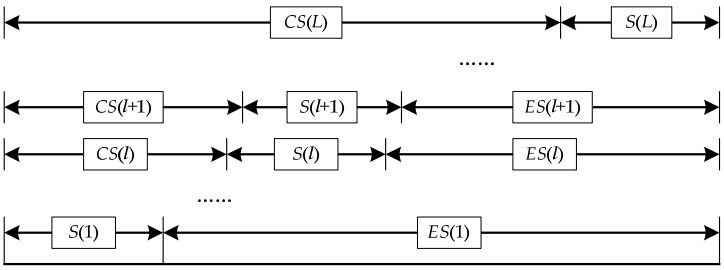

Figure 4. Operation-based rolling window.

4. Genetic Regulatory Network-Based Rescheduling Method

Based on the event-driven mechanism, a rescheduling problem is to be addressed if the delivery time deviation of jobs is larger than the deviation tolerance. A GRN-based method is proposed to solve this problem:

Step (1) Genes are generated to represent the decision variables. In terms of decision variables y_{rm} and x_{rki} in Table 1, two kinds of genes (i.e., $\{\boldsymbol{\pi}_{rm}|r = 1,2,\cdots,R;\ m = 1,2,\cdots,M\}$ and $\{\boldsymbol{\pi}_{rki}|r = 1,2,\cdots,R;\ k = 1,2,\cdots,K_i;\ i = 1,2,\cdots,I\}$) are generated. The gene $\boldsymbol{\pi}_{rm}$ denotes that the rth job entering the HFS belongs to job type m, whereas the gene $\boldsymbol{\pi}_{rki}$ denotes that the rth job entering the HFS is processed on the kth machine at the ith production stage.

Step (2) Regulation equations are developed to describe the constraints and objectives:

$$\boldsymbol{\sigma}_z = f_z(\boldsymbol{\mu}_1(n), \boldsymbol{\mu}_2(n), \cdots, \boldsymbol{\mu}_Z(n), \vartheta_1, \vartheta_2, \cdots, \vartheta_E), \tag{3}$$

where Z represents the number of genes in a GRN, $\boldsymbol{\mu}_z(n)$ is a binary variable that is equal to 1 if gene $\boldsymbol{\pi}_z$ is in the expressed state at the nth iteration, otherwise, $\boldsymbol{\mu}_z(n)$ is equal to 0, $\boldsymbol{\sigma}_z$ is the inhibition coefficient that describes the inhibitory effects on gene $\boldsymbol{\pi}_z$ quantitatively; $\vartheta_1, \vartheta_2, \cdots, \vartheta_E$ are regulatory parameters; and $f_z : \mathrm{R}^{(Z+E)} \rightarrow \mathbf{R}$ is a nonlinear function related to workshop conditions.

Step (3) Gene expression procedures are designed to determine solutions. At the beginning of such a procedure, the set of related genes is first confirmed based on operations within the rolling window. If a reactive scheduling is necessary, all these genes are initialized to the unexpressed state. At each iteration (i.e., $n = 1, 2, 3, \cdots, N$), some of these genes are converted to the expressed state based on the regulation equations. When $n > N$, genes in the expressed state are confirmed, and their corresponding decision variable values are equal to 1 in the rescheduling solution.

Step (4) Regulatory parameters are optimized to minimize the makespan. A near-optimal solution is obtained by gene states $\{y_{rm}|r = 1,2,\cdots,R; m = 1,2,\cdots,M\}$ and $\{x_{rki}|r = 1, 2, \cdots, R;\ k = 1, 2, \cdots, K_i;\ i = 1, 2, \cdots, I\}$ that are decided by the optimized regulation equations.

As shown in Figure 5, the regulation equation and expression procedure of genes $\{\pi_{rm}|r = 1, 2, \cdots, R;\ m = 1, 2, \cdots, M\}$ and those of genes $\{\boldsymbol{\pi}_{rki}|r = 1, 2, \cdots, R;\ k = 1, 2, \cdots, K_i;\ i = 1, 2, \cdots, I\}$ are presented in Sections 4.1 and 4.2, respectively. The regulatory parameter optimization procedure is given in Section 4.3.

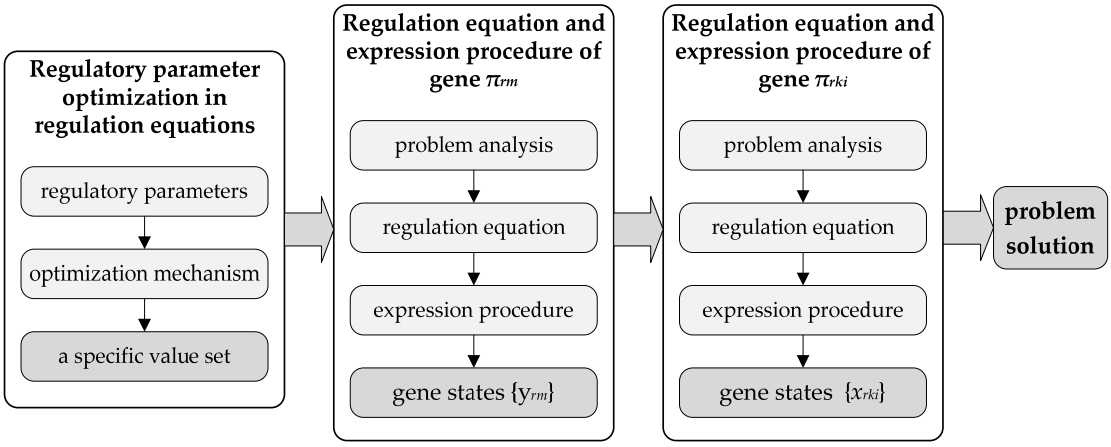

Figure 5. Outline of the GRN-based method.

4.1. Regulation Equation and Expression Procedure of Gene $\boldsymbol{\pi}_{rm}$

In HFS scheduling, each machine prefers to operate on the same job type in order to reduce the setup time. For instance, the jobs of type m_1 are operated on the 1st machine (i.e., the jobs on this machine are thus "m_1-m_1-m_1-m_1- ... "), and those of m_2 and m_3 are assigned to the 2nd machine and the 3rd machine, respectively. In this case, the job sequence entering this production stage will be "m_1-m_2-m_3-m_1-m_2-m_3- ... " because the parallel machines perform their tasks simultaneously. Thereupon, each job type would appear in the job sequence cyclically, and the related cycle time should be accorded with the number of parallel machines at each stage. However, the greedy nature of rule-based sequencing methods will probably keep the HFS scheduling choosing the "easy" jobs, which means jobs of a "hard" type will be left to the last positions of the job sequence. For instance, four jobs of type m_4 might be left for the last four positions of a job sequence, which is not preferred because there is no scheduling flexibility when assigning these jobs to parallel machines. On the contrary, it is much better to leave four jobs m_1, m_2, m_3 and m_4 for remaining positions because the job assigning procedure can choose machines with shorter processing time, less changeovers or earlier availability for each job to minimize the makespan further. For this reason, it is necessary to keep the stable production rate of each job type in the job sequence in order to avoid the near sightedness of rule-based sequencing methods. In terms of these facts, following rules should be obeyed:

1. A job type cannot be selected if its cycle of entering the HFS is not accord with the number of parallel machines at each stage;
2. A job type cannot be selected if its production ratio differs from its demand ratio.

No model sequence could satisfy all these rules completely, and each unsatisfied case might increase the makespan. The regulation equation of gene $\boldsymbol{\pi}_{rm}$ is thus developed as follows:

$$u_{rm} = \varepsilon_1 \sum_{i=1}^{I} \left| \frac{\sum_{v=1}^{r-1} y_{vm} + 1}{r} - \frac{1}{K_i} \right| + \varepsilon_2 \left| \frac{\sum_{v=1}^{r-1} y_{vm} + 1}{r} - \frac{d_{lm}}{D_l} \right| + H\left(\sum_{v=W_l}^{r-1} y_{vm} - d_{lm} \right) + H\left(\sum_{m=1}^{M} y_{rm} \right), \quad (4)$$

where u_{rm} represents the inhibition coefficient to gene $\boldsymbol{\pi}_{rm}$, d_{lm} represents the number of job type m entering the HFS within the lth window, $D_l = \sum_{m=1}^{M} d_{lm}$, W_l represents the number of jobs having entered the HFS before the lth window, ε_1 and ε_2 are regulatory parameters, and $H(x)$ is a step function that satisfies $H(x) = 0\,(x \leq 0)$ and $H(x) = +\infty\,(x > 0)$. The first two terms of the right side of Equation (4) represent the inhibition strength to gene $\boldsymbol{\pi}_{rm}$ owing to Rules 1 and 2, respectively. The last two terms ensure the job sequence to satisfy the predetermined quantity of each job type.

In the expression procedure of gene $\boldsymbol{\pi}_{rm}$, all the operations are arranged in an ascending sequence in terms of their starting time in the predictive plan. p operations with their starting time later than

the rescheduling instant are selected consecutively in the operation sequence. Assuming that d_{lm} represents the number of operations processed on the first production stage for job type m, there are $D_l = \sum_{m=1}^{M} d_{lm}$ jobs to be arranged for the job sequence within the lth window (from the W_lth position to the $(W_l + D_l)$th position). The genes $\{\boldsymbol{\pi}_{rm} | r = W_l, W_l + 1, \cdots, W_l + D_l;\ m = 1, 2, \cdots, M\}$ are thus initialized to the unexpressed state in the rescheduling problem, whereas the gene states $\{y_{rm} | r = 1, 2, \cdots, W_l;\ m = 1, 2, \cdots, M\}$ and $\{y_{rm} | r = W_l + D_l + 1, W_l + D_l + 2, \cdots, R;\ m = 1, 2, \cdots, M\}$ are given values based on the predictive plan. The gene expression procedure deals with the gene states $\{y_{rm} | r = W_l, W_l + 1, \cdots, W_l + D_l;\ m = 1, 2, \cdots, M\}$. At each iteration $n = \boldsymbol{\alpha}$ ($\boldsymbol{\alpha} \in \{1, 2, \cdots, D_l\}$), the inhibition coefficient u_{rm} is calculated for genes $\left\{\boldsymbol{\pi}_{(W_l + \boldsymbol{\alpha})m} | m = 1, 2, \cdots, M\right\}$ and the gene with minimum u_{rm} is converted to the expressed state. When $n > D_l$, the job sequence within the lth window is rescheduled based on $\{y_{rm} | r = W_l, W_l + 1, \cdots, W_l + D_l;\ m = 1, 2, \cdots, M\}$. Appendix A presents this gene expression procedure within the lth window.

4.2. Regulation Equation and Expression Procedure of Gene $\boldsymbol{\pi}_{rki}$

The major objective of assigning jobs to alternative machines is to avoid machine idle time because the makespan is decreased mainly by increasing the utilization of machines. For this reason, the rule of assigning each waiting processing job to the earliest available machine is widely adopted. In addition, each machine prefers to operate on the same job type in order to reduce changeover activities while setup times are taken into consideration. Therefore, assigning a job to parallel machines at each stage should comply with following rules:

1. A machine cannot be selected if the waiting time of a job on this machine is longer than that on another machine.
2. A machine cannot be selected if it requires setup time for a job.

It is almost impossible to satisfy these rules completely, and each unsatisfied case might increase the objective function value. In addition, the 5th assumption should also be obeyed. The regulation equation of gene $\boldsymbol{\pi}_{rki}$ is thereby developed as follows:

$$w_{rki} = h_1 \sum_{v=1}^{K_i} (a_{rki} - a_{rvi}) + h_2 c_{b_{rki} m_0 ki} + H(\sum_{v=1}^{K_i} x_{rvi}), \tag{5}$$

where w_{rki} represents the inhibition coefficient to gene $\boldsymbol{\pi}_{rki}$, a_{rki} and b_{rki} represent the end time and the job type of last operation on the kth parallel machine when the rth job enters the ith production stage, m_0 represents the type of the rth job (i.e., $m_0 = \sum_{m=1}^{M} m y_{rm}$), h_1 and h_2 are regulatory parameters, and $H(x)$ is a step function that satisfies $H(x) = 0$ $(x \leq 0)$ and $H(x) = +\infty$ $(x > 0)$. The first two terms of the right side of Equation (5) represent the inhibition strength to gene $\boldsymbol{\pi}_{rki}$ owing to Rules 1 and 2, respectively. The last term describes the constraint originated from the 5th assumption. The variable b_{rki} satisfies:

$$b_{rki} = \begin{cases} b_{(r-1)ki} & \textbf{if} \quad x_{(r-1)ki} = 0 \\ \sum_{m=1}^{M} m y_{(r-1)m} & \textbf{if} \quad x_{(r-1)ki} = 1 \end{cases}, \tag{6}$$

and the variable a_{rki} satisfies:

$$a_{rki} = \begin{cases} a_{(r-1)ki} & \textbf{if} \quad x_{(r-1)ki} = 0 \\ a_{(r-1)ki} + c_{b_{(r-1)ki} m_0 ki} + t_{m_0 ki} & \textbf{if} \quad x_{(r-1)ki} = 1 \ \textbf{and} \ a_{rki} \geq \sum_{k=1}^{K_{i-1}} a_{rk(i-1)} x_{rk(i-1)} \\ \sum_{k=1}^{K_{i-1}} a_{rk(i-1)} x_{rk(i-1)} + c_{b_{(r-1)ki} m_0 ki} + t_{m_0 ki} & \textbf{if} \quad x_{(r-1)ki} = 1 \ \textbf{and} \ a_{rki} < \sum_{k=1}^{K_{i-1}} a_{rk(i-1)} x_{rk(i-1)} \end{cases}, \tag{7}$$

Assuming φ_{li} jobs (from the $\boldsymbol{\sigma}_{li}$th position to the $(\boldsymbol{\sigma}_{li} + \varphi_{li})$th position in the job sequence) enter the ith production stage based on the $p = \sum_{i=1}^{I} \varphi_{li}$ operations selected for the lth window, the genes $\left\{\boldsymbol{\pi}_{(\boldsymbol{\sigma}_{li} + \boldsymbol{\beta}_i)ki} | \boldsymbol{\beta}_i = 1, 2, \cdots, \varphi_{li};\ k = 1, 2, \cdots, K_i;\ i = 1, 2, \cdots, I\right\}$ are initialized to the unexpressed state, whereas other gene states are in accordance with the predictive plan. In the expression procedure

of gene $\boldsymbol{\pi}_{rki}$, at each iteration $n = \boldsymbol{\beta}_i + \sum_{v=1}^{i-1} \varphi_{lv}$ ($i = 1, 2, \cdots, I$, $\boldsymbol{\beta}_i \in \{1, 2, \cdots, \varphi_{li}\}$), the inhibition coefficient w_{rki} is calculated for genes $\left\{\boldsymbol{\pi}_{(\boldsymbol{\sigma}_{li}+\boldsymbol{\beta}_i)ki} | k = 1, 2, \cdots, K_i\right\}$ and the gene with minimum w_{rki} is converted to the expressed state. When $n > p$, the production plan within the lth window is rescheduled based on gene states $\left\{x_{(\boldsymbol{\sigma}_{li}+\boldsymbol{\beta}_i)ki} | \boldsymbol{\beta}_i = 1, 2, \cdots, \varphi_{li};\ k = 1, 2, \cdots, K_i;\ i = 1, 2, \cdots, I\right\}$. The pseudo codes of this procedure are presented in Appendix B.

4.3. Regulatory Parameter Optimization

Based on values of regulatory parameters $\boldsymbol{\varepsilon}_1$ and $\boldsymbol{\varepsilon}_2$, the gene expression procedure governed by Equation (4) determines the gene states $\{y_{rm} | r = 1, 2, \cdots, R;\ m = 1, 2, \cdots, M\}$ within the lth window, each of which represents whether the rth job entering the HFS belongs to job type m. Moreover, the regulatory parameters h_1 and h_2 specify the gene regulation in Equation (5) and further determine the gene states $\{x_{rki} | r = 1, 2, \cdots, R;\ k = 1, 2, \cdots, K_i;\ i = 1, 2, \cdots, I\}$ within the lth window. Each gene state x_{rki} represents whether the rth job is manufactured on the kth machine at the ith production stage. In this way, a solution to the rescheduling problem is obtained based on regulatory parameter values (i.e., $\boldsymbol{\varepsilon}_1$, $\boldsymbol{\varepsilon}_2$, h_1 and h_2). Figure 6 illustrates this procedure in the rescheduling of a specific HFS.

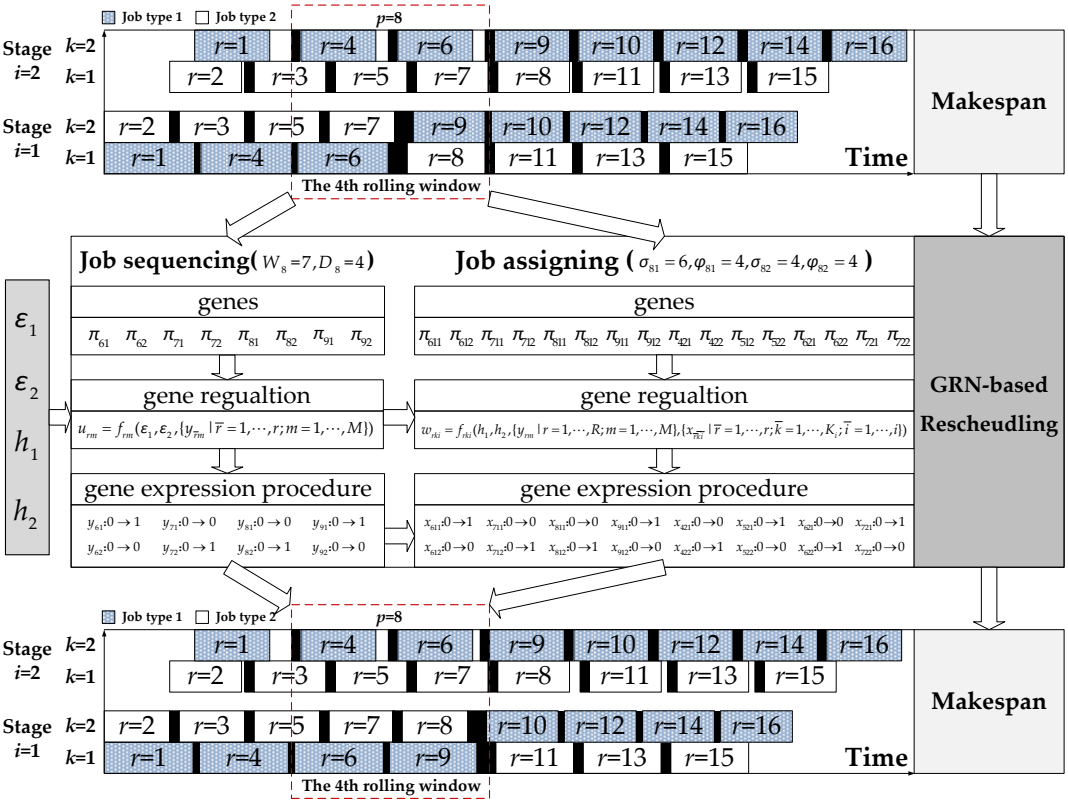

Figure 6. An example of GRN-based rescheduling solution (M = 2, I = 2, k_1 = 2, k_2 = 2, R = 16, d_1 = 8, d_2 = 8).

A parameter optimization procedure is further implemented to minimize the makespan. According to the regulation equations, a gene receives inhibition if its expression breaks scheduling rules, and the inhibition strength is weighted by related regulatory parameters. These parameters have different values because each rule plays its distinctive role in determining an optimal solution. For example, decreasing setup times is important if machines require comparatively longer durations for job type changeovers, and the parameter h_2 should have a large value. The sensitivity analysis on makespan can evaluate the importance weights and thus realize regulatory parameter optimization [42]. Alternatively, machining learning methods can also optimize these parameters if there is enough

historical data. For instance, a neural network with workshop conditions can be trained to recommend appropriate parameters.

Apart from these analytical and machine learning methods, random searching algorithms (e.g., genetic algorithms and immune learning algorithms) are also alternative to optimize parameters [43]. For instance, each individual in the genetic algorithm can represent a specific value set of regulatory parameters, and the fitness value of this individual can be evaluated based on related makespan. Through a series of genetic operations (i.e., evaluation, crossover and mutation), the best individual in this algorithm determines the minimum makespan. Moreover, a real-coded algorithm should be used in the parameter optimization procedure because the regulatory parameters are real variables.

5. Numerical Results

5.1. Strategy Parameter Analysis

As discussed in Section 3, delivery deviation tolerance and operation-based window size are important parameters in the event-driven rescheduling strategy. To determine appropriate parameter values, the numeric tests listed in Table 2 are presented.

Table 2. Strategy parameter analysis experiment.

Problem Parameter	Numerical Range
Number of production stages (I)	4
Number of parallel machines at each stage (K_i)	U[2, 4]
Number of job types (M)	4
Production volume of each job type (d_m)	8
Processing time (t_{mki}) (s)	U[20, 30]
Changeover time between same job types (c_{mmki}) (s)	U[1, 3]
Changeover time between different job types ($c_{mm'ki}$) (s)	U[5, 7.5]

For these numeric tests, a static scheduling result is first obtained by taking the whole planning horizon as a special rescheduling window and using the GRN-based method. A real-coded genetic algorithm is specifically used in this method to optimize regulatory parameters, in which the population size is 200, the maximum generation is 50, the crossover possibility is 0.8, and the mutation possibility is 0.1. Figure 7 illustrates the Gantt chart of static scheduling results (makespan = 392 s). In this diagram, white rectangles represent processing times and black ones denote changeover times. Moreover, the numbers within each rectangle represents the batch of a job type. For instance, "1, 3" represents the third batch of the first job type.

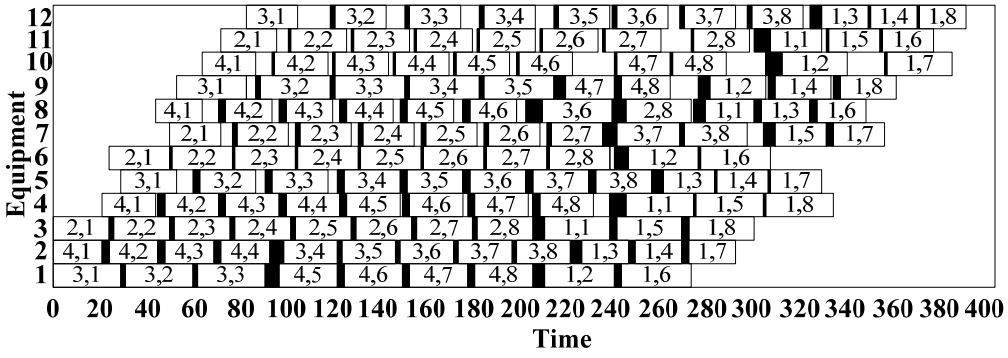

Figure 7. Gantt chart of a static scheduling.

Assuming that the processing times follow a 4-order Erlang distribution, the rescheduling strategies with different deviation tolerances and window sizes are then used. Table 3 lists the makespan, rescheduling times and computational time of strategies with different deviation tolerances while the window size is 20 operations. In this table, the strategy with the delivery time deviation tolerance of 0.125 minimizes the makespan. Table 4 thus lists dynamic scheduling results with different window sizes while the delivery time deviation tolerance is 0.125. Based on these results, the delivery deviation tolerance of 0.125 and the window size of 77 operations are adopted to generate dynamic scheduling results, as shown in Figure 8.

Table 3. Results of delivery time deviation tolerance experiments.

Delivery Time Deviation Tolerance	Makespan (s)	Rescheduling Times	Computational Time (ms)
0.025	408.08	39	65,708
0.05	400.55	15	23,842
0.075	401.03	7	11,346
0.1	407.20	5	7245
0.125	394.34	2	3243
0.15	394.93	2	3193
0.2	395.22	1	1602
0.25	395.22	1	1602
0.3	439.46	0	16
0.35	439.46	0	19
0.4	439.46	0	18
0.45	439.46	0	21
0.5	439.46	0	19

Table 4. Results of rolling window size experiments.

Rolling Window Size (Operation)	Makespan (s)	Rescheduling Times	Computational Time (ms)
13	394.54	2	3216
26	397.55	3	4997
39	400.78	4	7914
51	394.56	4	9305
64	398.19	3	7219
77	386.78	4	13,006
90	390.33	2	6363
102	393.33	2	7519
115	389.03	4	18,545
128	390.54	2	10,400

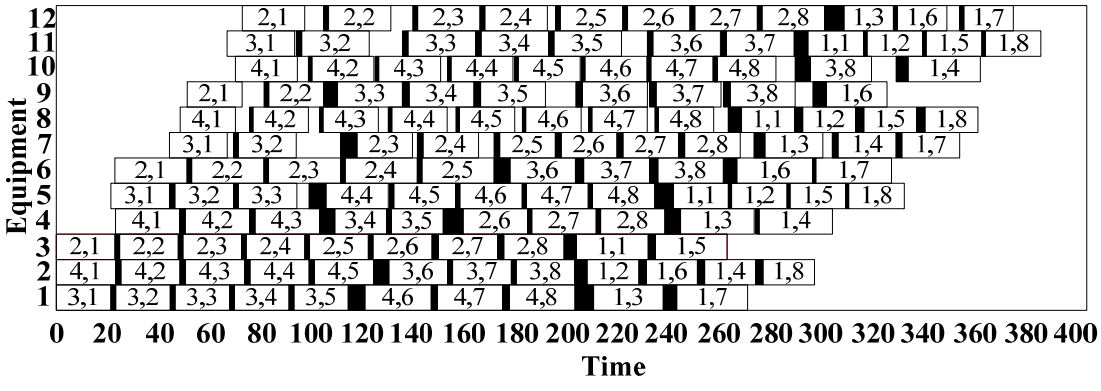

Figure 8. Gantt chart of a dynamic scheduling.

5.2. Comparative Experiments

Nine benchmarks from Qin et al. [44] are further used to validate the effectiveness of the GRN-based method, as shown in Table 5. Assuming that actual processing times follow the Erlang distribution, these benchmarks are solved by the GRN-based method and the Improved Ant Colony Algorithm (IACO) introduced in [44], respectively. An Intel® Core™ i7-2720QM CPU @ 2.20 GHz and 8.00 GB RAM based notebook computer (Dell Inc., Xiamen, China) is adopted to test these experiments. Table 6 lists the experimental results. Because both the IACO and the GRN-based method are based on random search procedures, all these results are averaged values over 20 replications.

Table 5. Nine benchmarks for dynamic HFS (Hybrid Flow Shop) scheduling.

Benchmark	Number of Jobs	Number of Stages	Number of Machines at Each Stage	Processing Times (s)	Setup Times (s) (Same Jobs)	Setup Times (s) (Different Jobs)
6 × 2	6	2	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
30 × 2	30	2	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
100 × 2	100	2	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
6 × 4	6	4	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
30 × 4	30	4	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
100 × 4	100	4	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
6 × 8	6	8	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
30 × 8	30	8	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]
100 × 8	100	8	U[1, 5]	U[50, 70]	U[3, 5]	U[12, 24]

Table 6. Results of the GRN (Genetic Regulatory Network)-based method and IACO (Improved Ant Colony Algorithm) method.

Benchmark Name	GRN-Based Method			IACO Method		
	Makespan (s)	Rescheduling Times	CPU Time (ms)	Makespan (s)	Rescheduling Times	CPU Time (ms)
6 × 2	256.27	2.3	13.3	252.79	1.15	164.5
30 × 2	573.58	4.25	48.6	568.45	1.6	580.3
100 × 2	886.23	1.35	354.5	855.01	1.9	8130.5
6 × 4	1071.56	1.2	26.0	1098.29	1.85	365.2
30 × 4	2249.04	2.35	67.5	2314.21	1.2	3906.2
100 × 4	2532.37	3.15	498.4	2761.85	1.35	33,538.3
6 × 8	3471.06	1.3	51.7	3536.51	1.55	921.4
30 × 8	6859.97	1.9	97.6	6978.14	1.9	13,641.4
100 × 8	7396.41	5.1	3055.2	7646.71	1.85	21,9613.3

As shown in Table 6, the IACO method achieves better results than the GRN-based methods for the benchmarks "6 × 2", "30 × 2" and "100 × 2". Because these benchmarks are featured with a small-scale solution space, the IACO method is possibly to search out the optimal solutions via its global searching procedure, whereas the GRN-based method might fail to find an optimal one owing to the predetermined rules embedded in its regulation equations. When the problem scale increases in the benchmarks "6 × 4" to "100 × 8", the GRN-based method obtains smaller makespans than the IACO method. The IACO can hardly search out optimal solutions, or even near-optimal ones, when the solution space is increased, whereas the GRN-based method ensures a good solution owing to its embedded rules and can further find a better one by optimizing regulatory parameters. Moreover, the GRN-based method optimizes four parameters, rather than all the decision variables in the IACO method. This proposed method can thus save the CPU time and demonstrate better response capability to processing time variations in HFS.

Thereupon, the GRN-based method is validated as an effective and efficient method for dynamic scheduling in HFS.

5.3. Case Study

A specific PCB assembly shop composed of four production stages (SMT chip processing stage, plug processing stage, welding processing stage, and test stage) are also investigated. At each production stage, several production lines are alternative for PCB assembly (S1, S2 and S3 at the 1st stage; M1 and M2 at the 2nd stage; A1 and A2 at the 3rd stage; and T1 and T2 at the 4th stage). Ten types of PCBs are assembled in these lines. The requirement for each PCB type is 1000. The processing times and setup times are listed in Tables 7 and 8, respectively.

Table 7. Processing times (s) of different PCB (Printed Circuit Board) types in each production line.

PCB Type	S1	S2	S3	M1	M2	A1	A2	T1	T2
3ET0321AF	4.615	1	1.2	0.3	0.24	0.6	0.75	0.6	0.6
3ET0322AF	4.615	1	0.293	0.4	0.3	0.6	0.75	0.6	0.55
3ET0100CET	1	1	0.6	0.06	1.5	0.75	0.75	0.4	0.55
3ET0141CET	1	1.428	1	0.4	0.24	0.6	0.3	0.55	0.5
3ET0349CET	1	1	0.6	0.4	0.3	0.75	0.75	0.6	0.6
3ET0630CET	1	1	1	0.06	0.24	0.75	0.6	0.55	0.5
3ET0631CET	1	1	0.6	0.06	0.3	0.6	0.75	0.55	0.4
3ET0741CET	1	4.615	0.293	0.6	0.3	0.75	0.6	0.4	0.4
3ET0374TEK	1.428	0.923	1.2	0.4	0.3	0.6	0.3	0.6	0.55
3ET0435TEK	0.923	0.923	0.882	0.6	1.5	0.6	0.4	0.6	0.4

Table 8. Setup times (s) between PCBs.

PCB Type	3ET0 321AF	3ET0 322AF	3ET0 100CET	3ET0 141CET	3ET0 349CET	3ET0 630CET	3ET0 631CET	3ET0 741CET	3ET0 374TEK	3ET0 435TEK
3ET0321AF	1.56	12.58	8.37	13.12	10.83	7.28	8.37	7.1	9.51	9.2
3ET0322AF	5.73	1.32	9.64	7.3	9.53	10.57	5.77	10.67	12.13	9.82
3ET0100CET	9.81	7.23	1.23	6.34	11.22	14.63	12.2	11.09	12.98	6.54
3ET0141CET	8.93	8.27	11.62	1.28	12.96	5.97	8.6	5.69	9.95	8.02
3ET0349CET	14.12	10.73	10.05	7.79	1.32	14.26	9.83	14.6	7.07	9.29
3ET0630CET	8.93	10.83	10.36	10.79	14.15	1.56	14.95	11.8	10.88	10.12
3ET0631CET	9.14	12.32	14.64	9.99	9.37	7.51	1.75	10.37	13.43	5.48
3ET0741CET	12.19	9.81	14.22	8.99	9.52	14.33	14.43	1.22	14.09	11.83
3ET0374TEK	13.17	7.75	14.34	10.12	7.54	5.85	8.44	6.82	1.38	5.03
3ET0435TEK	6.44	7.89	12.52	13.34	10.28	13.01	10.72	6.14	7.85	1.52

Figure 9 illustrates the Gantt chart of static scheduling results obtained by using the GRN-based method. As shown in this figure, all PCB products are divided into 100 batches, and the makespan is 3308 s. Because the starting time of operations is directly determined by the end time of the former operation on the same machine or that of the same job, the processing time variations have cumulative impacts on the completion time of jobs. Based on static scheduling results, the IACO method is used to deal with processing time variations. Figure 10 is the Gantt chart obtained by the IACO method, in which the makespan is 3415.77 s. The GRN-based method is also used to solve the dynamic scheduling problem and achieves a makespan of 3261.98 s, as shown in Figure 11. By integrating job sequencing rules and job assigning rules in a reasonable manner, the GRN-based method ensures waiting jobs to be in-time assigned to an idle machine with comparatively shorter setup time. The regulatory optimization procedure realizes the tradeoff between shorter job waiting time and less changeover activities to minimize the makespan. However, the IACO method fails to realize these targets for some rolling windows because its global search procedure can hardly search out optimal solutions for a real-time scheduling. Taking the first production stage for example, the Garnet chart in Figure 10 realizes higher utilization of Machine 1 and shorter job waiting time on Machine 3 than that in Figure 11. Consequently, the GRN-based method is a more effective scheduling method than the IACO method.

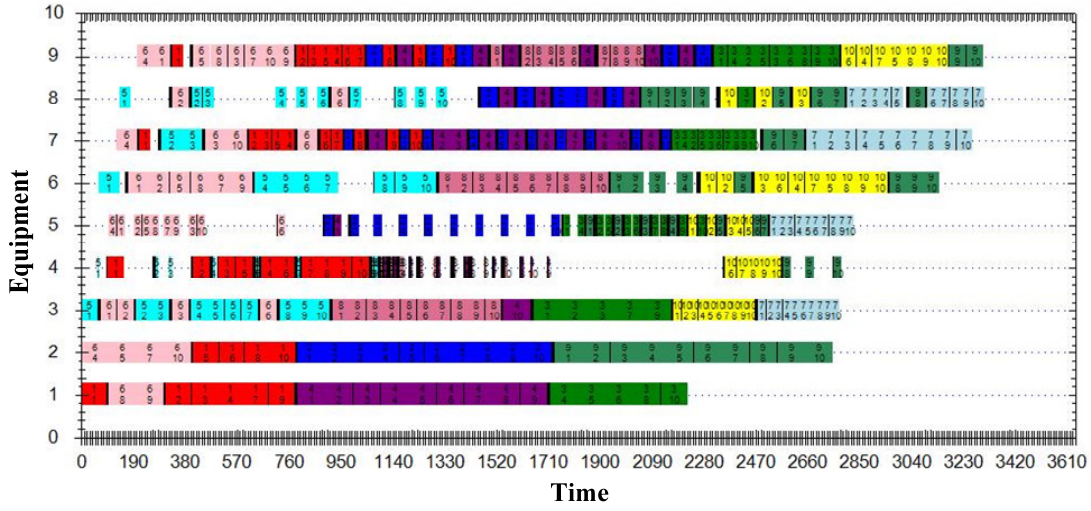

Figure 9. Static scheduling Gantt chart.

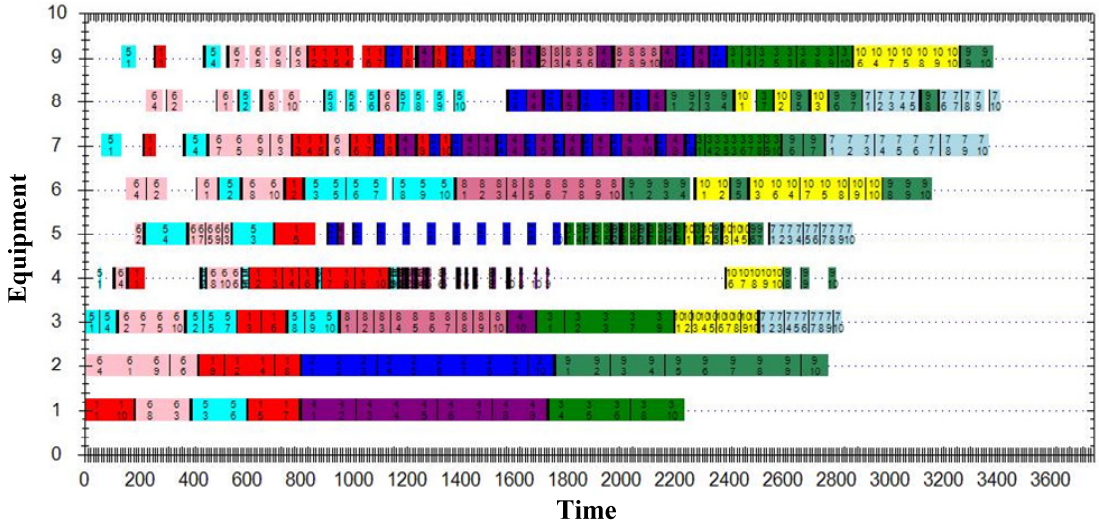

Figure 10. IACO (Improved Ant Colony Algorithm) dynamic scheduling Gantt chart.

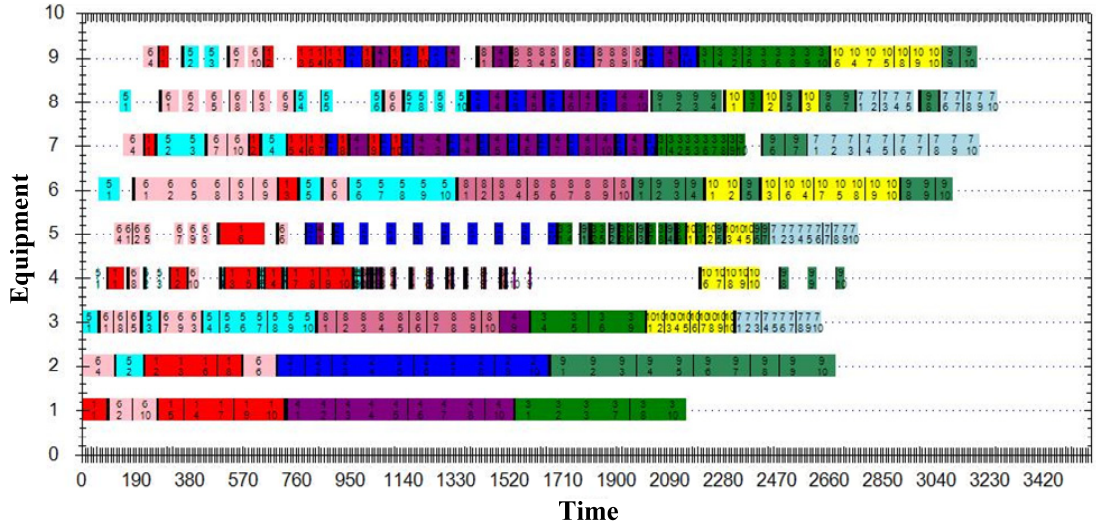

Figure 11. GRN-based dynamic scheduling Gantt chart.

6. Conclusions

This paper solves the HFS scheduling problem with uncertain processing times based on the predictive-reactive strategy. For the rescheduling problems in response to processing time variation events, a novel GRN-based method is developed to minimize the makespan. The critical factor is the employment of GRN to describe the HFS scheduling problem and some scheduling rules. This enables the regulatory parameter optimization procedure to generate near-optimal rescheduling solutions within the reasonable computational time. The effectiveness of this method over the IACO method was demonstrated by a series of benchmark tests and the case study in a PCB assembly shop.

This paper investigates the dynamic event of processing time variations in HFS scheduling, however, the machine breakdowns are also common in real environments. These events will cause the machines to be unavailable for a certain duration known as Mean Time To Repair (MTTR). To deal with these situations, the dynamic scheduling should first generate a predictive plan with minimum makespan based on the assumptions that all the machines are reliable and that the processing times take their expectation values. The job delivery deviation caused by dynamic events of machine breakdowns as well as processing time variations is then monitored during the execution of this plan. If the deviation is beyond a tolerance, the GRN-based method reschedules the production plan within a rolling window to respond to these events. This method will have very complex regulation equations because the unavailability of failed machines needs to be involved in the calculation of job waiting time and machine idle time. Thereupon, the HFS scheduling will be more challenging when the dynamic events of machine breakdowns are taken into account and it will also have better practical values at the same time. Therefore, in our further work, we will develop an enhanced rescheduling method that deals with the dynamic events of machine breakdowns as well as processing time variations by using the GRN. In addition, we will extend this method to multi-objective scheduling that minimizes the due date of jobs, the idle time of machines, the scheduling adjustment cost, etc.

Acknowledgments: The authors would like to acknowledge the financial support of the National Natural Science Foundation of China (No. 51435009; No. 51275307).

Author Contributions: Youlong Lv and Jie Zhang proposed the method and wrote the paper; Wei Qin conceived and designed the experiments; and Youlong Lv and Wei Qin performed the experiments and analyzed the data.

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

Appendix A

Pseudo codes of expression procedure of gene $\boldsymbol{\pi}_{rm}$:

```
//initialization of genes related to the (l + 1) th window
for α ← 1 to D_l do
    for m ← 1 to M do
        r = W_l + α
        y_rm ← 0            //all genes are in the unexpressed state
    next;
next;

//expression circulation
for α ← 1 to D_l do
    m_0 ← 1, u_0 ← +∞
    r = W_l + α             //current iteration
    for m ← 1 to M do
        calculate u_rm in Equation (4)        //inhibition coefficient of current gene
        if u_rm < u_0 then
            m_0 ← m           //update index of the gene with minimum u_rm
            u_0 ← u_rm        //update the minimum u_rm
        end if;
    next;
    y_rm0 ← 1          //convert the gene with minimum u_rm to the expressed state
next.
```

Appendix B

Pseudo codes of expression procedure of gene π_{rki}:

```
//initialization of genes related to the (l + 1)th window
for i ← 1 to I do
    for r ← σ_li to σ_li + φ_li do
        for k ← 1 to K_i do

            x_rki ← 0            //all genes are in the unexpressed state
        next;
    next;
next;

//real-time shop information
for i ← 1 to I do
    for k ← 1 to K_i do
        r = σ_li
        get a_rki and b_rki
    next;
next;

//expression circulation
for i ← 1 to I do
    for r ← σ_li to σ_li + φ_li do
        k_0 ← 1, w_0 ← +∞
        for k ← 1 to K_i do          //current iteration
            calculate w_rki in Equation (4)          //inhibition coefficient of current gene
            if w_rki < w_0 then
                k_0 ← k              //index of the gene with minimum w_rki
                w_0 ← w_rki          //the minimum w_rki
            end if;
        next;
        x_rk0i ← 1          //convert the gene with minimum w_rki to the expressed state
        r = r + 1
        for k ← 1 to K_i do
            calculate a_rki in Equation (5)      //update shop information
            calculate b_rki in Equation (6)      //update shop information
        next;
    next;
next.
```

References

1. Gholami, M.; Zandieh, M.; Alem-Tabriz, A. Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *Int. J. Adv. Manuf. Technol.* **2009**, *42*, 189–201. [CrossRef]
2. Chou, F. Der Particle swarm optimization with cocktail decoding method for hybrid flow shop scheduling problems with multiprocessor tasks. *Int. J. Prod. Econ.* **2013**, *141*, 137–145. [CrossRef]
3. Yan, H.S.; Jiang, T.H.; Xiong, F.L. A hybrid electromagnetism-like algorithm for a mixed-model assembly line sequencing problem. *Int. J. Ind. Eng. Theory Appl. Pract.* **2014**, *21*, 153–167.
4. Arnaout, J.P.; Rabadi, G.; Musa, R. A two-stage Ant Colony Optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *J. Intell. Manuf.* **2010**, *21*, 693–701. [CrossRef]
5. Lamothe, J.; Marmier, F.; Dupuy, M.; Gaborit, P.; Dupont, L. Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints. *Comput. Oper. Res.* **2012**, *39*, 1236–1244. [CrossRef]
6. Grabowski, J.; Pempera, J. Sequencing of jobs in some production system. *Eur. J. Oper. Res.* **2000**, *125*, 535–550. [CrossRef]

7. Sawik, T.J. Mixed integer programming for scheduling surface mount technology lines. *Int. J. Prod. Res.* **2001**, *39*, 3219–3235. [CrossRef]
8. Wong, T.C.; Chan, F.T.S.; Chan, L.Y. A resource-constrained assembly job shop scheduling problem with Lot Streaming technique. *Comput. Ind. Eng.* **2009**, *57*, 983–995. [CrossRef]
9. Yalaoui, N.; Ouazene, Y.; Yalaoui, F.; Amodeo, L.; Mahdi, H. Fuzzy-metaheuristic methods to solve a hybrid flow shop scheduling problem with pre-assignment. *Int. J. Prod. Res.* **2013**, *51*, 3609–3624. [CrossRef]
10. Salvador, M.S. A solution to a special class of flow shop scheduling problems. In *Symposium on the Theory of Scheduling and Its Applications*, 1st ed.; Elmaghraby, S.E., Ed.; Springer: Berlin, Germany, 1973; pp. 83–91.
11. Yan, H.S.; Xia, Q.F.; Zhu, M.R.; Liu, X.L.; Guo, Z.M. Integrated Production Planning and Scheduling on Automobile Assembly Lines. *IIE Trans.* **2003**, *35*, 711–725. [CrossRef]
12. Sun, Y.; Zhang, C.; Gao, L.; Wang, X. Multi-objective optimization algorithms for flow shop scheduling problem: A review and prospects. *Int. J. Adv. Manuf. Technol.* **2011**, *55*, 723–739. [CrossRef]
13. Rossi, A.; Puppato, A.; Lanzetta, M. Heuristics for scheduling a two-stage hybrid flow shop with parallel batching machines: Application at a hospital sterilisation plant. *Int. J. Prod. Res.* **2013**, *51*, 2363–2376. [CrossRef]
14. Chan, F.T.S.; Wang, Z. Robust production control policy for a multiple machines and multiple product-types manufacturing system with inventory inaccuracy. *Int. J. Prod. Res.* **2014**, *52*, 4803–4819. [CrossRef]
15. Kis, T.; Pesch, E. A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *Eur. J. Oper. Res.* **2005**, *164*, 592–608. [CrossRef]
16. Ruiz, R.; Vázquez-rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [CrossRef]
17. Pan, Q.K.; Wang, L.; Gao, L. A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers. *Appl. Soft Comput.* **2011**, *11*, 5270–5280. [CrossRef]
18. Wang, H.; Chou, F.; Wu, F. A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan. *Int. J. Adv. Manuf. Technol.* **2011**, *53*, 761–776. [CrossRef]
19. Yang, J. Minimizing total completion time in a two-stage hybrid flow shop with dedicated machines at the first stage. *Comput. Oper. Res.* **2015**, *58*, 1–8. [CrossRef]
20. Mirsanei, H.S.; Zandieh, M.; Moayed, M.J.; Khabbazi, M.R. A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. *J. Intell. Manuf.* **2011**, *22*, 965–978. [CrossRef]
21. Wang, I.L.; Yang, T.; Chang, Y.B. Scheduling two-stage hybrid flow shops with parallel batch, release time, and machine eligibility constraints. *J. Intell. Manuf.* **2012**, *23*, 2271–2280. [CrossRef]
22. Wang, S.; Liu, M.; Chu, C. A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. *Int. J. Prod. Res.* **2015**, *53*, 1143–1167. [CrossRef]
23. Komaki, G.M.; Teymourian, E.; Kayvanfar, V. Minimising makespan in the two-stage assembly hybrid flow shop scheduling problem using artificial immune systems. *Int. J. Prod. Res.* **2015**, *7543*, 1–21. [CrossRef]
24. Sun, L.; Yu, S. Scheduling a real-world hybrid flow shop with variable processing times using Lagrangian relaxation. *Int. J. Adv. Manuf. Technol.* **2015**, *78*, 1961–1970. [CrossRef]
25. Tang, L.; Liu, W.; Liu, J. A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. *J. Intell. Manuf.* **2005**, *16*, 361–370. [CrossRef]
26. Chryssolouris, G. *Manufacturing Systems: Theory and Practice*, 2nd ed.; Springer Science & Business Media: New York, NY, USA, 2006; pp. 465–597.
27. Mehta, S.V.; Uzsoy, R.M. Predictable scheduling of a job shop subject to breakdowns. *IEEE Trans. Robot. Autom.* **1998**, *14*, 365–378. [CrossRef]
28. Yao, F.S.; Zhao, M.; Zhang, H. Two-stage hybrid flow shop scheduling with dynamic job arrivals. *Comput. Oper. Res.* **2012**, *39*, 1701–1712. [CrossRef]
29. Feng, X.; Zheng, F.; Xu, Y. Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times. *Int. J. Prod. Res.* **2016**, *54*, 3706–3717. [CrossRef]
30. Chryssolouris, G.; Velusamy, S. Dynamic scheduling of manufacturing job shops using genetic algorithms. *J. Intell. Manuf.* **2001**, *12*, 281–293. [CrossRef]
31. Gourgand, M.; Grangeon, N.; Norre, S. A review of the static stochastic flow-shop scheduling problem. *J. Decis. Syst.* **2000**, *9*, 1–31. [CrossRef]

32. Joo, B.J.; Choi, Y.C.; Xirouchakis, P. Dispatching rule-based algorithms for a dynamic flexible flow shop scheduling problem with time-dependent process defect rate and quality feedback. *Procedia CIRP* **2013**, *7*, 163–168. [CrossRef]
33. Wang, K.; Choi, S.H.; Qin, H.; Huang, Y. A cluster-based scheduling model using SPT and SA for dynamic hybrid flow shop problems. *Int. J. Adv. Manuf. Technol.* **2013**, *67*, 2243–2258. [CrossRef]
34. Chryssolouris, G.; Dicke, K.; Lee, M. An approach to real-time flexible scheduling. *Int. J. Flex. Manuf. Syst.* **1994**, *6*, 235–253. [CrossRef]
35. Rahmani, D.; Ramezanian, R. A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Comput. Ind. Eng.* **2016**, *98*, 360–372. [CrossRef]
36. Michalos, G.; Makris, S.; Mourtzis, D. A web based tool for dynamic job rotation scheduling using multiple criteria. *CIRP Ann. Manuf. Technol.* **2011**, *60*, 453–456. [CrossRef]
37. Sahin, F.; Narayanan, A.; Robinson, E.P. Rolling horizon planning in supply chains: Review, implications and directions for future research. *Int. J. Prod. Res.* **2013**, *51*, 5413–5436. [CrossRef]
38. Sun, Y.; Feng, G.; Cao, J. Stochastic stability of Markovian switching genetic regulatory networks. *Phys. Lett. A* **2009**, *373*, 1646–1652. [CrossRef]
39. Jong, H.D. Modeling and simulation of genetic regulatory systems: A literature review. *J. Comput. Biol.* **2002**, *9*, 67–103. [CrossRef] [PubMed]
40. Wang, Y.; Wang, Z.; Liang, J.; Li, Y.; Du, M. Synchronization of stochastic genetic oscillator networks with time delays and Markovian jumping parameters. *Neurocomputing* **2010**, *73*, 2532–2539. [CrossRef]
41. Wahde, M.; Hertz, J. Coarse-grained reverse engineering of genetic regulatory networks. *Biosystems* **2000**, *55*, 129–136. [CrossRef]
42. Shan, R.; Zhao, Z.S.; Chen, P.F.; Liu, W.J.; Xiao, S.Y.; Hou, Y.H.; Wang, Z. Network modeling and assessment of ecosystem health by a multi-population swarm optimized neural network ensemble. *Appl. Sci.* **2016**, *6*, 175. [CrossRef]
43. Quan, G.Z.; Zhang, Z.H.; Zhang, L.; Liu, Q. Numerical Descriptions of Hot Flow Behaviors across β Transus for as-Forged Ti–10V–2Fe–3Al Alloy by LHS-SVR and GA-SVR and Improvement in Forming Simulation Accuracy. *Appl. Sci.* **2016**, *8*, 210. [CrossRef]
44. Qin, W.; Zhang, J.; Song, D. An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time. *J. Intell. Manuf.* **2015**. [CrossRef]