Journal of Intelligent Manufacturing, 13, 339–351, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

Robust design of flexible manufacturing systems using, colored Petri net and genetic algorithm

KAZUHIRO SAITOU, SAMIR MALPATHAK and HELGE QVAM

Department of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan, 48109-2125, USA
E-mail: kazu@umich.edu

Received April and accepted November 2001

A method is presented for the robust design of flexible manufacturing systems (FMS) that undergo the forecasted product plan variations. The resource allocation and the operation schedule of a FMS are modeled as a colored Petri net and an associated transition firing sequence. The robust design of the colored Petri net model is formulated as a multi-objective optimization problem that simultaneously minimizes the production costs under multiple production plans (batch sizes for all jobs), and the reconfiguration cost due to production plan changes. A genetic algorithm, coupled with the shortest imminent operation time (SIO) dispatching rule, is used to simultaneously find the near-optimal resource allocation and the event-driven schedule of a colored Petri net. The resulting Petri net is then compared with the Petri nets optimized for a particular production plan in order to address the effectiveness of the robustness optimization. The simulation results suggest that the proposed robustness optimization scheme should be considered when the products are moderately different in their job specifications so that optimizing for a particular production plan creates inevitably bottlenecks in product flow and/or deadlock under other production plans.

Keywords: Flexible manufacturing systems, robust design, colored Petri nets, genetic algorithms, part families.

1. Introduction

Flexible manufacturing systems (FMS) are a class of manufacturing systems that can be quickly configured to produce multiple types of products (jobs). The adoption of an FMS, over a dedicated manufacturing system (DMS), is often motivated by the need of agile manufacturing that can quickly adopt changes in production plans (batch sizes for all jobs) due to market demand fluctuations. While the increased flexibility of FMS could provide greater overall productivity under various production scenarios, it imposes increased complexity in the allocation of available resources to different operations required in making each product, and the scheduling of the sequence of activities to accomplish the best production efficiency (Lee and DiCeasre, 1994).

In order to quickly adapt to fluctuating market demands, the resource allocation and the scheduling—referred to as configuration in this paper—of an FMS should not simply be optimized for the current production plan. Rather, it should ideally be optimized for robustness against the variation in production plans, so that the system can deal with the variation with minimal reconfiguration while achieving consistently efficient production under all production plans of interest (Saitou and Malpathak, 1999; Saitou and Quam, 1998)

Assuming the forecasts of production plan variations are available, let us consider a scenario where an FMS simultaneously produces two types of products A and B with various fractions while the total number of production per unit time (e.g., a day) is kept constant. When A and B are very similar in their job specifications, then, it is conjectured that one would not need to consider robustness optimization since the

configuration optimized for the current production plan is robust enough such that little system reconfigurations are necessary to deal with production plan changes (imagine the extreme of this case where A and B are identical). On the other hand, when products under simultaneous production are moderately different, a slight change in the production plan will heavily impact the production efficiency, possibly due to the creation of bottlenecks in the product flow. This would necessitate the system reconfiguration in order to achieve the efficient production under the new production plan.

The aforementioned conjecture motivated us to develop a methodology for robustness optimization of FMS that underwent forecasted product plan variations, and to study the effectiveness of the methodology in the simultaneous production of products with various similarities. Viewing the demand fluctuation as a uncontrollable factor, the robustness of FMS design is defined as the ability to achieve consistently a high production efficiency with the minimum system reconfiguration, regardless of the variations in the production plan within a forecasted range. A configuration of an FMS is modeled as a colored Petri net and an associated transition firing sequence. The robustness optimization of the colored Petri net model is formulated as a multi-objective optimization problem that minimizes the production costs under multiple production plans, and the reconfiguration cost due to production-plan changes. A genetic algorithm, coupled with the SIO (shortest imminent operation time) dispatching rule, is used to simultaneously find the near-optimal resource (machine) allocation and the event-driven schedule of a colored Petri net. The resulting Petri nets are then compared with the Petri nets optimized for a particular production plan in order to validate the above conjecture.

2. Related work

Petri nets (Petri, 1962) have been used for the analysis and simulation of FMS due to their capability of modeling the concurrency, synchronization and sequencing in discrete-event systems (Dubios and Stecke, 1983; Narahari and Viswanadham; Zhu and DiCesare). In addition to the use as an analysis tool, Petri net models are often used for FMS scheduling problems. Given a job specification (operation sequences needed for each job, the machine types and processing time for each operation), and the corresponding resource allocation (the number of machines in each type), one can construct a Petri net model of an FMS, where the event-driven operation schedules of the modeled FMS are represented as the transition firing sequences of the Petri net. Due to the NP-completeness of the underlying job-shop scheduling problem (JSSP) (Garey and Johnson, 1979) a near-optimal schedule is often found via heuristic search algorithms such as beam search (Shin and Sekiguchi, 1991), A* algorithm (Lee and DiCeasre, 1993) and genetic algorithms (Chiu and Fu, 1997), coupled with the discrete-event simulation of the operation of the Petri net model.

In general, the quality of the optimal schedule is influenced by the quality of the resource allocation (i.e., the topology of the Petri net model) for a given job specification. This motivates the simultaneous optimization of resource allocation and scheduling, a generalization of JSSP known as generalized resource-constrained project scheduling problems (GRCPSP), which is also NP-complete (Garey and Johnson, 1979). GRCPSP is typically formulated as discrete programming problems and solved by heuristic search algorithms (Sprecher, 1994). The solution provides a near-optimal allocation of a given resources (i.e., machines) and time-driven operation schedules. Although event-driven schedules are often preferred for FMS scheduling due to their robustness (Lee and Dicesare, 1933), discrete-event-based models such as Petri nets are rarely used for GRCPSP due to the computational time for the model simulation.

Recently, several researchers attempted the use of robust design methods in flexible manufacturing systems. Shang applied Taguchi's signal-to-noise concept to design material handling systems in FMS, using the mean time between failure and the mean time to repair as noise factors. Bulgak *et al.* (1999) proposed a similar method for asynchronous flexible assembly systems subject to stochastic jam rates and jam clear times. While these work, focus on the robust resource allocation for a given operation schedule, other work deals with the problem of the robust operation of an FMS for a given resource allocation (Jafari, 1992; Wu, 1999; Stecke and Raman, 1996). Viewing the demand fluctuation of a part mix as uncontrollable inputs to an FMS, Askin *et al.* (1997) discussed the methodology of flexible

cellular manufacturing system design using an interactive cell formation method. The present work also assumes the demand variation as an uncontrollable factor, and attempts to achieve robust resource allocations and operation schedules that minimize overall production cost including reconfigurations.

3. Problem formulation

3.1. *Colored Petri net model of manufacturing systems*

Colored Petri nets (Alla *et al*., 1985; David and Alla, 1992) are an extension of ordinary Petri nets where a place can contain multiple tokens distinguished by a ''color'' associated with each token. As an ordinary Petri net, a colored Petri net is a directed graph consisting of two types of nodes, places and transitions. Two nodes are connected by a directed edge that connects either a place to a transition or a transition to a place (see Figs 1–3). Colored Petri net is adopted as a model of FMS since it allows the natural representations of the following issues critical to the present work:

- Event-driven schedules as transition firing sequences.
- Multiple product types as colors of tokens.
- System reconfigurations as topological changes of graphs.

In a basic form, a colored Petri net R is defined as a six-tuple:

$$R = \langle P, T, pre, post, m_0, C \rangle \qquad (1)$$

where P is a set of places, T is a set of transitions and C is a set of colors. *pre* and *post* are the functions of type $P \times T \times C \mapsto \mathbf{Z}^{|C|}$, and $m_0 : P \mapsto \mathbf{Z}^{|C|}$ is the initial marking, where $\mathbf{Z}$ is a set of integers. A place $p \in P$ is graphically represented by a circle, and a transition $t \in T$ is represented by a bar. A place can contain one or more tokens (with possibly different colors). The number and colors of tokens at a place $p \in P$ are called marking of the place denoted as $m(p)$, where $m : P \mapsto \mathbf{Z}^{|C|}$, and are represented graphically as colored dots in a circle. In most literature, however, a token is represented as $\langle c \rangle$, where c is a symbol representing the color of the token, as they are not normally printed in color. Let places p and q, and a transition t be connected by edges (p, t) and (t, q). The place p is called an input place of the transition t, and the place q is called an output place of the transition t. The marking of places change according to the following rules:

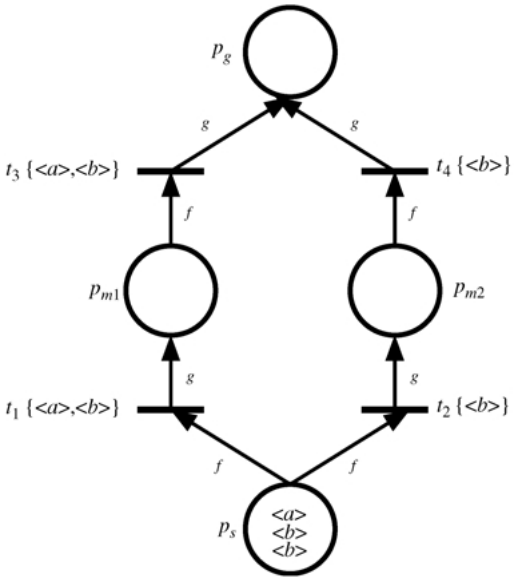

Fig. 1. Colored Petri net that models a production facility with start buffer p_s, two machines p_{m1} and p_{m2}, and goal buffer p_g with initial markings at clock = 0.

(1) For each input place p of a transition t, if $m(p) \leq pre(p, t, c)$; for a color, t is called enabled with respect to the color c.

(2) If a transition t is enabled with respect to a color c, it can fire.

(3) If a transition t enabled with respect to a color c fires, $m(p)$ changes to $m(p)$-$pre(p, t, c)$, and for each output place q of t, $m(q)$; changes to $m(q) + post(q, t, c)$.

In addition to the above basic definitions, the capacities and time associated with places are often defined in FMS modeling (timed places with capacities). In this case, an enabled transition t can fire only if an enabling token has been in the input place p longer than or equal to a specified time (this assumes a ''clock'' keeping track of the marking changes), and the total number of tokens does not exceed the capacity of the output place q as a result of marking change. A sequence of the marking changes in all places of a colored Petri net is called evolution of marking. The evolution of marking in a colored Petri net from the initial marking represents the

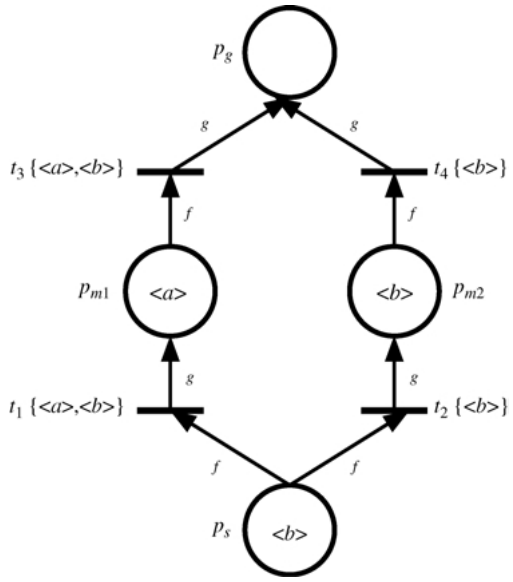

Fig. 2. State of the colored Petri net after firing of t_2 with $\langle b \rangle$ at clock = 1, and t_1 with $\langle a \rangle$ at clock = 2.

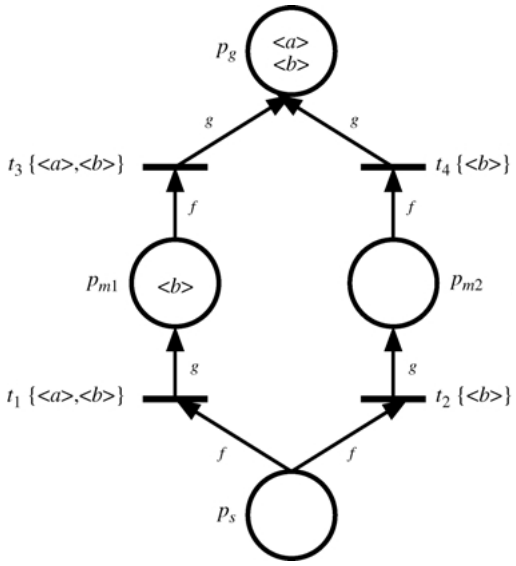

Fig. 3. State of the colored Petri net after firing of t_2 with $\langle b \rangle$ at clock = 1, t_1 with $\langle a \rangle$ at clock = 2, t_3 with $\langle a \rangle$ at clock = 3, t_1 with $\langle b \rangle$ at clock = 4, and t_4 with $\langle b \rangle$ at clock = 5.

sequence of event occurrences in the modeled discrete-event system.

Figures 1–3 illustrate the evolution of marking in a simple colored Petri net that models a production facility consisting of one ''start'' buffer p_s, and one machine p_{m1} of type M_1, and one machine p_{m2} of type M_2. The production facility is to produce two types of products $\langle a \rangle$ and $\langle b \rangle$, both of which need just one operation to finish. The machines of type M_1 are capable of performing this operation on both product types $\langle a \rangle$ and $\langle b \rangle$ with a unit time, while the machines of type M_2 can only perform the operation on product b with a unit time. This job specification is summarized in Table 1, where the columns indicate jobs (product types) and the rows indicate operations (only one for the both jobs in this example). The numbers in the parentheses adjacent to machine types M_1 and M_2 indicate the process time for the corresponding operation (all unity in this example). In this colored Petri net, $P = \{p_s, p_{m1}, p_{m2}, p_g\}$, $T = \{t_1, t_2, t_3, t_4\}$, $C = \{\langle a \rangle, \langle b \rangle\}$, $m_0(p_s) = (1,2)$, and $m_0(p_{m1}) = m_0(p_{m2}) = m_0(p_g) = (0,0)$. It is assumed that p_s and p_g have infinite capacities, and p_{m1} and p_{m2} have capacities equal to one (machines can process one product at a time). Since no products are either created or deleted during the operation of an FMS, functions *pre* and *post* are simply expressed in terms of ''shorthand'' functions $f, g : C \mapsto C$. For example,

$$pre(p_s, t_1, \langle a \rangle) = f(\langle a \rangle) = \langle a \rangle = (1,0)$$
$$post(p_{m1}, t_1, \langle a \rangle) = g(\langle a \rangle) = \langle a \rangle = (1,0)$$

The colors listed next to each transition in Figs 1–3 indicate the enabling colors of the transition, with which the transition can be enabled if appears in the input place.

At the start of the production cycle (i.e., clock = 0), the machines are not working and the unfinished products, one $\langle a \rangle$ and two $\langle b \rangle$s, are located in the start buffer p_s, as given in the initial marking $m_0(p_s) = (1,2)$. Since $m_0(p_s) > pre(p_s, t_1, \langle a \rangle)$, $pre(p_s, t_1, \langle b \rangle)$ and $m_0(p_s) > pre(p_s, t2, \langle b \rangle)$, transition t_1 is enabled with both $\langle a \rangle$ and $\langle b \rangle$, and t_2 is enabled with $\langle b \rangle$. Let us assume t_2 fires at the next clock cycle (clock = 1). Then, $m(p_s)$ changes from $(1,2)$ to $(1,1)$ as $pre(p_s, t_2, \langle b \rangle) = \langle b \rangle = (0,1)$, and hence one of two token $\langle b \rangle$s is removed from p_s. Also $m(p_{m2})$ changes from $(0,0)$ to $(0,1)$ as $post(p_{m2}, t_2, \langle b \rangle) = \langle b \rangle$, and hence the token $\langle b \rangle$

Table 1. Example job specification

	$J_{\langle a \rangle}$	$J_{\langle b \rangle}$
1	$M_1(1)$	$M_1(1)/M_2(1)$

removed form p_s appears in p_{m2}. At this point, transitions t_1, t_2 and t_4 are enabled. Let us assume t_1 fires at the next clock cycle (clock = 2). The transition t_1 has the choice of firing either $\langle a \rangle$ or $\langle b \rangle$. Suppose t_1 fires $\langle a \rangle$. Proceeding similar to the previous firing, the token $\langle a \rangle$ is removed from p_s and appears in p_{m1} (Fig. 2). This represents the system state of the machine p_{m1} processing the product $\langle a \rangle$ and the machine p_{m2} processing the product $\langle b \rangle$. In fact, p_{m2} has finished processing $\langle b \rangle$ at this point, but the completed $\langle b \rangle$ has not yet been transfered to p_g.

Although all transitions are enabled at this point, only t_3 or t_4 can fire since firing t_1 and t_2 would result in exceeding the capacity of places p_{m1} and p_{m2}. Let us assume t_3 is fired at the next clock cycle (clock = 3) and this moves $\langle a \rangle$ in p_{m1} to p_g. This means the machine p_{m2} has now finished processing the product $\langle a \rangle$ and sent it to the goal buffer p_g. Then, t_1, t_2, and t_4 are enabled but only t_1 or t_4 can fire. The subsequent firing of t_1 with $\langle b \rangle$ at clock = 4 followed by the firing of t_4 with $\langle b \rangle$ at clock = 5 would move $\langle b \rangle$ in p_g to p_{m1}, and $\langle b \rangle$ in p_{m2} to p_g (Fig. 3). This represents the machine p_{m1} is now processing product $\langle b \rangle$ while machine p_{m2} has finished processing product $\langle b \rangle$ and send it to the goal buffer.

At the next clock cycle (clock = 6), only t_1 can fire, which would move $\langle b \rangle$ in p_{m1} to p_g. One production cycle completes at this point since $m_0(p_s) = m(p_g)$, with the makespan being six clock cycles.

As illustrated above, a sequence of transition firing of a colored Petri net can be interpreted as an event-driven schedule of the modeled manufacturing systems. Therefore, choosing a transition firing sequence in the above example would result in a different evolution of markings, i.e., a different schedule, that would yield a different system behavior. In general, the topology of a colored Petri net model is determined by the job specification (operation sequences needed for each job, the machine types and processing time for each operation), and the corresponding resource allocation (the number of machines in each type).

3.2. *Robust design of FMS configurations*

We consider a scenario where an FMS simultaneously produces multiple types of products that share common resources. It is assumed that the production plan of the FMS is given as the batch sizes of all jobs, i.e., the numbers of each type of the products to be produced during a production cycle. Let n be the number of types of the products. Then, the production plan can be represented as $\rho \in \mathbf{Z}^n$. Suppose the total number of production (sum of the numbers of n product types) per unit time is kept constant to N, and hence the production plan changes are only due to the changes in the fraction of the product types. Let the fraction be α_i, where $0 \leq \alpha_i \leq 1$ for $i = 1, 2, \ldots, n$ and $\sum_{i=1}^{n} \alpha_i = 1$, or collectively be an n-dimensional vector $\mathbf{a}$. Given N, therefore, a production plan can be uniquely specified as a function of the fraction vector $\mathbf{a}$, which we shall call $\rho(\mathbf{a})$.

Let $\rho(\mathbf{a}_0)$ be the current production plan. We assume a forecast on the production plan changes within the timeframe of interest is available as a sequence of m production plans $\rho(\mathbf{a}_1), \rho(\mathbf{a}_2), \ldots, \rho(\mathbf{a}_m)$. Our objective is to optimize the robustness of the current configuration (resource allocation and schedule) of the FMS against the given variation in production plans. Namely, we want to minimize the reconfiguration while achieving consistently efficient production under all m production plans. Let $\mathbf{x}_0$ be the current configuration of the FMS, and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ be the future configurations corresponding to the forecasted m production plans. Then, the problem can be formulated as the simultaneous minimization of the following $2m + 2$ functions:

$$\mathit{makespan}(\mathbf{x}_j, \rho(\mathbf{a}_j)) \qquad j = 0, 1, \ldots, m \tag{2}$$

$$\mathit{facility\text{-}cost}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m) \tag{3}$$

$$\mathit{reconfig\text{-}cost}(\mathbf{x}_j, \mathbf{x}_{j+1}) \qquad j = 0, 1, \ldots, m-1 \tag{4}$$

where $\mathit{makespan}(\mathbf{x}_j, \rho(\mathbf{a}_j))$ is the makespan of the FMS with configuration $\mathbf{x}_j$ under production plan $\rho(\mathbf{a}_j)$, $\mathit{facility\text{-}cost}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m)$ is the total facility cost for configurations $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m$, and $\mathit{reconfig\text{-}cost}(\mathbf{x}_j, \mathbf{x}_{j+1})$ is the reconfiguration cost from configuration $\mathbf{x}_j$ to configuration $\mathbf{x}_{j+1}$.

Given configuration $\mathbf{x}_j$ and production plan $\rho(\mathbf{a}_j)$, $\mathit{makespan}(\mathbf{x}_j, \rho(\mathbf{a}_j))$ can be evaluated using a discrete-event simulation based on a colored Petri net model of an FMS. The facility cost is estimated simply as the total cost of the machines utilized in $m + 1$ configurations $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m$:

$$\mathit{facility\text{-}cost}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m) = \sum_k c_k \cdot \max_j \{n_k(\mathbf{x}_j)\} \tag{5}$$

where c_k is the cost of the machine of type k, and $n_k(\mathbf{x})$ is the number of the machines of type k used in configuration $\mathbf{x}$.

Reconfiguration cost from one configuration to the another is estimated as the number of rerouting required to accomplish the new configuration, i.e., the number of routings (connection between two places) in the colored Petri net that need to be changed due to the change in the resource allocation. The reconfiguration cost associated with the change in the schedules is not considered here since, as discussed in the following sections, the dynamic scheduling with dispatching rules adopted in this work achieves the schedule change with virtually no expenses, namely:

$$\textit{reconfig-cost}(\mathbf{x}_i, \mathbf{x}_j) = \text{number of routing differences from } \mathbf{x}_i \text{ to } \mathbf{x}_j \quad (6)$$

For instance, the reconfiguration cost from the colored Petri net in Fig. 4 to the one in Fig. 5 is 4 since two routings between p_s and p_g must be removed and added due to the removal of one machine p_{m2} of the type M_2 (which can only process $\langle b \rangle$), and the addition of second machine q_{m1} of the type M_1 (which can process $\langle a \rangle$ and $\langle b \rangle$).

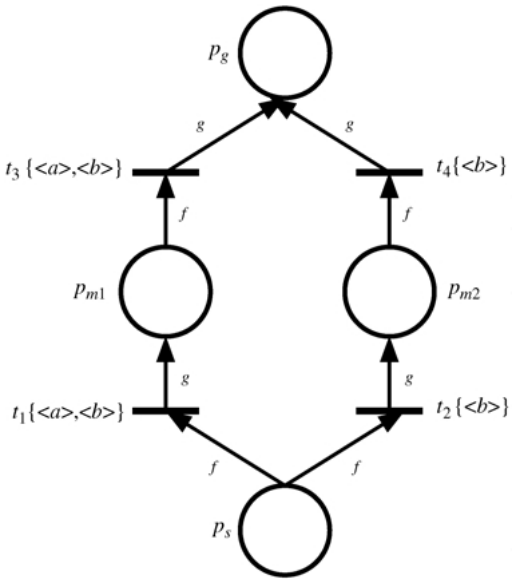

Fig. 4. A FMS before reconfiguration.

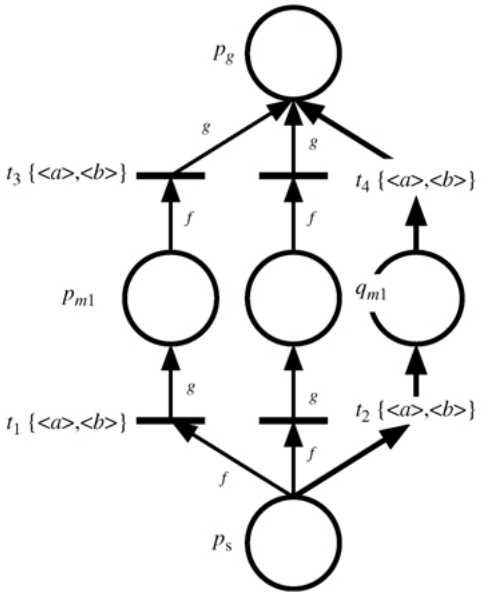

Fig. 5. A FMS after reconfiguration.

3.3. *Optimization using a genetic algorithm and dispatching rules*

The robustness optimization of FMS configurations discussed in the previous section requires the simultaneous optimization of resource allocation and scheduling. Due to the high complexity of the underlying optimization problem (GRCPSP), a hybrid scheme is adopted where a genetic algorithm is used for resource allocation, and dispatching rules are used for dynamic scheduling of the colored Petri net models. Although the resulting configuration is not guaranteed to be optimal, this hybrid scheme allows very fast evaluation of a large number of feasible configurations. With the dynamic scheduling, however, even a small change in the resource allocation might cause a large change in scheduling, resulting in a large change in the production cost. This type of behavior often gives difficulty in optimization algorithms that rely on the local landscape of the objective function, which necessitates the use of an intelligent ''generate-and-test'' type of algorithms such as a genetic algorithm.

Genetic algorithms (GAs) are an optimization technique in which the points in design space are analogous to the organisms subject to a process of natural selection, or ''survival of the fittest'' (Goldberg, 1989; Holland, 1975). At an iteration

(often called generation in GA), the quality of a chromosome, a bit string representation of a point in the design space, is measured based on a fitness function, and highly-fit chromosomes have higher chances to be selected for reproduction. Two ''parent'' chromosomes selected for reproduction are mated through genetic crossover, resulting in two offsprings that are likely to inherit good ''genes'' from their parents. Many generations of such selection and mating will produce a highly-fit population of chromosomes, i.e., better designs.

Dispatching rules are the local rules that specify priorities in the dispatching of products to machines while production is in progress. Dispatching rules have been traditionally used for scheduling, due to their simplicity and reliability. A number of dispatching rules such as First-In-First-Out (FIFO), SIO, and shortest remaining processing time (SRPT) has been successfully applied to FMS scheduling (Choi and Malstrom, 1988). Although the schedules created by off-line algorithms often outperform the ones by dispatching rules, they allow very fast and dynamic creation of near-optimal schedules. Also, the schedules created by dispatching rules tend to be robust against the sudden change in the resource allocation (e.g. machine breakdown), since the schedules are dynamically created during the operation, rather than determined off-line. In this work, the SIO rule is used for dynamic scheduling of a colored Petri net, whose resource allocation is specified by a ''chromosome,'' of a genetic algorithm.

4. Simulation results

This section describes the case studies of the examples with $n=2$ and $m=1$. In other words, two product types A and B are to be produced, and only one forecast on the production plan is available.

4.1. *Assumptions*

The job specification that characterizes a product type is specified as the numbers of operations needed to complete the product, the machine types capable of performing each operation, and their process times. In the following examples, the job specifications of two product types are represented as a table (Table 2) similar to Table 1. It is assumed that all machine types have a capacity of one, i.e., a machine can process only one product at a time.

Since $n=2$, faction vector $\mathbf{a}$ has dimension 2, and hence can be expressed using one parameter α as $(\alpha, 1-\alpha)$, where $0 \le \alpha \le 1$. Unless otherwise specified, the total number of production per cycle $N=20$, the current production plans is $\alpha=0.9$ (i.e., 18 As and 2 Bs), and the forecasted production plan is $\alpha=0.1$ (i.e., 2 As and 18 Bs). The $2m+2$ functions as defined above with $m=1$ are aggregated as a weighted sum:

$$\begin{aligned} f(\mathbf{x}_0, \mathbf{x}_1) = {} & w_m \cdot \{makespan(\mathbf{x}_0, \rho(\alpha_0)) \\ & + makespan(\mathbf{x}_1, \rho(\alpha_1))\} \\ & + w_f \cdot facility\text{-}cost(\mathbf{x}_0, \mathbf{x}_1) \\ & + w_r \cdot reconfig\text{-}cost(\mathbf{x}_0, \mathbf{x}_1) \end{aligned} \quad (7)$$

where $\mathbf{x}_0$ and $\mathbf{x}_1$ are the configurations for current production plan $\rho(\alpha_0)$ and forecasted production plan $\rho(\alpha_1)$, and w_m, w_f and w_r are the weights of the makespan, the facility cost and the reconfiguration cost, respectively.

In order to study the effectiveness of the robustness optimization, in each example the resulting optimal configuration pair $(\mathbf{x}_0^*, \mathbf{x}_1^*)$ is compared with two configurations: the one optimized only for the current production plan, and the one optimized *only* for the forecasted production plan. We shall refer to these two configurations as $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$, respectively. The comparison is done by plotting the values of the following three functions representing the production cost of each configuration, evaluated with α varying between $\alpha=0$ (only A produced) and $\alpha=1$ (only B produced):

$$\begin{aligned} & w_m \cdot \min\{makespan(\mathbf{x}_0^*, \rho(\alpha)); makespan(\mathbf{x}_1^*, \rho(\alpha))\} \\ & \quad + w_f \cdot facility\text{-}cost(\mathbf{x}_0^*, \mathbf{x}_1^*) \end{aligned} \quad (8)$$

$$w_m \cdot makespan(\tilde{\mathbf{x}}_0, \rho(\alpha)) + w_f \cdot \sum_k c_k \cdot n_k(\tilde{\mathbf{x}}_0) \quad (9)$$

$$w_m \cdot makespan(\tilde{\mathbf{x}}_1, \rho(\alpha)) + w_f \cdot \sum_k c_k \cdot n_k(\tilde{\mathbf{x}}_1) \quad (10)$$

We shall refer to this plot as the production cost–alpha plot. In the production cost–alpha plot, the production cost of the optimal configuration pair $(\mathbf{x}_0^*, \mathbf{x}_1^*)$ is denoted as Petri net 3, and the production costs of $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$ are denoted as Petri net 1 and Petri net 2, respectively. Note that Petri net 3 actually consists of two Petri nets defined by $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$—the minimum

production cost of these two Petri nets is plotted in the production cost–alpha plot.

The results in the following examples are obtained by a steady-state GA with the population size 100, the number of generations 50, the probability of crossover 0.9 and the probability of mutation 0.05. The discrete-event simulation code is written in C++, and GALib from MIT CADLAB with some in-house enhancements is used as an optimizer. The optimization runs took at most five minutes with a 300-MHz Sun UltraSPARC 10 Workstation.

4.2 Example 1

The first example is on the production scenario where both jobs $J_{\langle a\rangle}$ and $J_{\langle b\rangle}$ require two operations, and there are three machine types M_1, M_2, and M_3 available, as shown in Table 2. The job specification indicates that the machine type 2 is the only resource shared between two jobs. This can be more clearly seen by observing the topology of the colored Petri net model in the case when there exists only one machine for all machine types, which we shall refer to as the basic Petri net.

Figure 6 shows the basic Petri net of this job specifications, where p_b is a buffer (with an infinite capacity) that stores both product types after the completion of the first operation. During the optimization process, the Petri net corresponding to a particular resource allocation is constructed by adding or removing machines of each type in this basic Petri net. It is assumed that the total number of the machines is four, consisting of no more than three machines per each machine type. During optimization, a penalty is imposed to the objective function (7) proportional to the amount of violation of these bounds. In addition, the costs of all machine types are assumed to be one.

Table 3 shows the optimal resource allocations for different weights $\mathbf{w} = (w_m, w_f, w_r)$, where $\mathbf{n}(\mathbf{x}) = (n_1(\mathbf{x}), n_2(\mathbf{x}), n_3(\mathbf{x}))$ denotes the vector of the numbers of machine types M_1, M_2 and M_3 in the

Table 2. Job specification for Example 1

	$J_{\langle a\rangle}$	$J_{\langle b\rangle}$
1	$M_2(9)/M_3(3)$	$M_1(2)/M_2(6)$
2	$M_3(4)$	$M_2(7)$

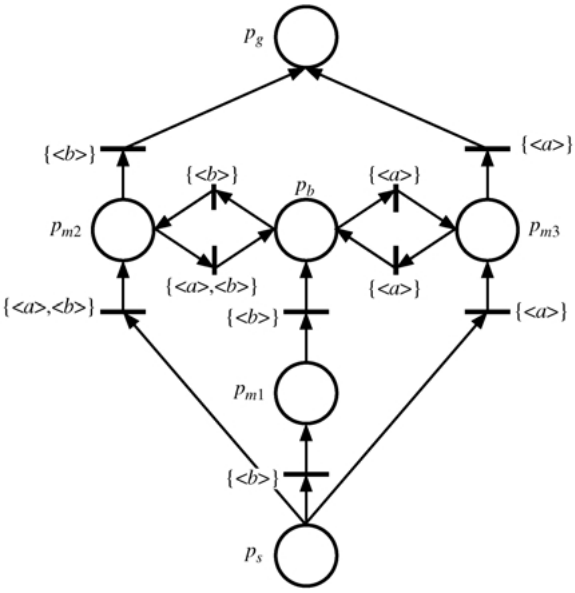

Fig. 6. Basic Petri net of the job specifications in Table 2. During the optimization process, the Petri net corresponding to a particular resource allocation is constructed by adding or removing machines of each type in this basic Petri net.

configuration $\mathbf{x}$. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0, 1, 3)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0, 3, 1)$. Since they are quite different each other $(\textit{reconfig-cost}(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1) = 16)$, the robustness optimization cannot simply converge to $\mathbf{x}_0^* = \tilde{\mathbf{x}}_0$ and $\mathbf{x}_1^* = \tilde{\mathbf{x}}_1$ even with $w_r = 1$, and forced to find ''compromised'' solutions, as shown in the second row of Table 3. It is also observed that $\mathbf{n}(\mathbf{x}_0^*)$ and $\mathbf{n}(\mathbf{x}_1^*)$ become closer as w_r increases. For $w_r = 5$ (the bottom row of Table 3), in fact, $\mathbf{n}(\mathbf{x}_0^*)$ and $\mathbf{n}(\mathbf{x}_1^*)$ converged to an identical value $(\textit{reconfig-cost} = 0)$, which is quite different from both $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$.

Figures 7–9 show the production cost–alpha plot for the cases corresponding to each row of Table 3, i.e., $\mathbf{w} = (1, 1, 1)$, $(1, 1, 3)$ and $(1, 1, 5)$, respectively. The ''switch'' between $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ occurs at $\alpha = 0.7$ and $\alpha = 0.5$ in Figs 7 and 8, respectively. Since for $w_r = 1$ the optimizer is not strongly forced to find $\mathbf{x}_0^*$

Table 3. Resource allocation result for Example 1. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0, 1, 3)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0, 3, 1)$

$\mathbf{w}$	$\mathbf{n}(\mathbf{x}_0^*)$	$\mathbf{n}(\mathbf{x}_1^*)$	*reconfig-cost*
$(1, 1, 1)$	$(0, 1, 3)$	$(1, 2, 1)$	14
$(1, 1, 3)$	$(0, 2, 2)$	$(1, 2, 1)$	6
$(1, 1, 5)$	$(0, 2, 2)$	$(0, 2, 2)$	0

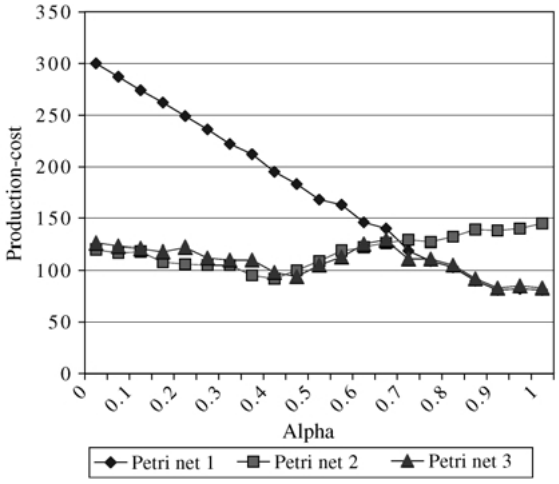

Fig. 7. Production cost–alpha plot for Example 1 with $w_r = 1$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0, 1, 3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0, 3, 1)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (0, 1, 3)$ and $\mathbf{n}(\mathbf{x}_1^*) = (1, 2, 1)$.

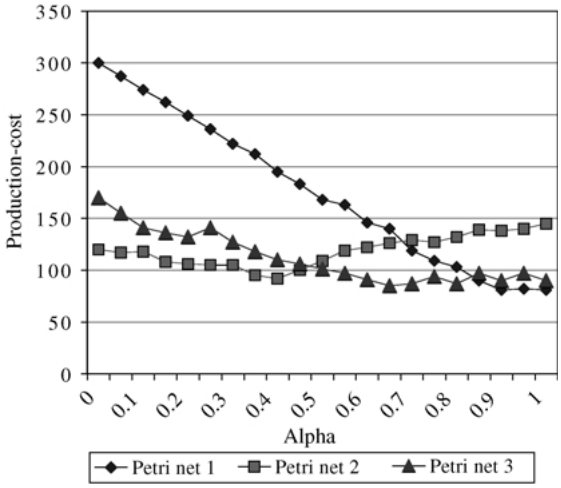

Fig. 9. Production cost–alpha plot for Example 1 with $w_r = 5$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0, 1, 3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0, 3, 1)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (0, 2, 2)$ and $\mathbf{n}(\mathbf{x}_1^*) = (0, 2, 2)$.

and $\mathbf{x}_1^*$ that are close to each other, $\mathbf{x}_0^*$ tends to be close to $\tilde{\mathbf{x}}_0$ (in fact equal in this case), and $\mathbf{x}_1^*$ tends to be close to $\tilde{\mathbf{x}}_1$. This results in the production cost of Petri net 3 over the range of $0 \leq \alpha \leq 1$ being quite similar to the minimum of the ones of Petri net 1 and Petri net 2, as shown in Fig. 7.

As w_r increases, $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ are forced to be closer. This results in the solutions whose production costs are not as good as $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$ at $\alpha = 0.9$ and $\alpha = 0.1$, respectively, but consistently low (in other words, *robust*) over a wide range of α. This trend is clearly shown in the case of $w_r = 3$ (Fig. 8) and $w_r = 5$ (Fig. 9). Although converged to one configuration, the overall production cost of Petri net 3 for $w_r = 5$ is higher than the production cost of the one for $w_r = 3$, especially for $0 \leq \alpha \leq 0.5$. In other words, $\mathbf{x}_0^* = \mathbf{x}_1^*$ is achieved with the price of higher production costs.

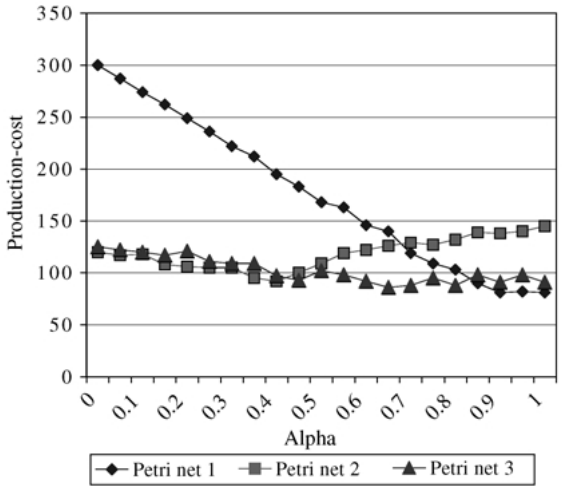

Fig. 8. Production cost–alpha plot for Example 1 with $w_r = 3$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0, 1, 3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0, 3, 1)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (0, 2, 2)$ and $\mathbf{n}(\mathbf{x}_1^*) = (1, 2, 1)$.

As seen in the above results, the robustness optimization was quite effective in this example. It is observed that the limited resource sharing in the job specifications causes $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_0$ to be quite different. Since they are quite different, the production flow bottlenecks are quickly created when running $\tilde{\mathbf{x}}_0$ with $\alpha < 0.9$ or running $\tilde{\mathbf{x}}_1$ with $\alpha > 0.1$, which causes the significant increase in makespan. This is more evident in Petri net 1 that has to rely on only one M_2 to process $J_{\langle b \rangle}$, which is quite slow. In such cases, the robustness optimization seems effectively find the configuration pair that exhibits the robust performances over a range of α.

4.3. *Example 2*

The second example is taken from Example 2 of the work by Lee and DiCesare. This example is similar to Example 1 with more extensive resource sharing among jobs. The job specification shown in Table 4 indicates all machine types M_1, M_2 and M_3 are shared between two jobs $J_{\langle a \rangle}$ and $J_{\langle b \rangle}$. This extensive resource sharing creates complex routings among machines

Table 4. Job specification for Example 2

	$J_{\langle a\rangle}$	$J_{\langle b\rangle}$
1	$M_2(17)/M_3(11)$	$M_1(10)/M_2(14)$
2	$M_1(10)/M_3(18)$	$M_2(20)/M_3(10)$

Table 5. Resource allocation result for Example 2. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,0,3)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,0,2)$

w	$\mathbf{n}(\mathbf{x}_0^*)$	$\mathbf{n}(\mathbf{x}_1^*)$	*reconfig-cost*
$(1,1,1)$	$(1,0,3)$	$(2,0,2)$	8
$(1,1,3)$	$(2,0,2)$	$(2,0,2)$	0

and a buffer, as illustrated in the basic Petri net shown in Fig. 10. In the figure, p_b is a buffer (with an infinite capacity) that stores both product types after the completion of the first operation. As in Example 1, the total number of the machines is bounded to four, and no more than three machines are allowed for each machine type. The costs of all machine types are assumed to be one.

Table 5 shows the optimal resource allocations for different weights $\mathbf{w} = (w_m, w_f, w_r)$. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,0,3)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,0,2)$. Since they are fairly close, ($reconfig\text{-}cost(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1) = 8$), the robustness optimization can converge to $\mathbf{x}_0^* = \tilde{\mathbf{x}}_0$ and $\mathbf{x}_1^* = \tilde{\mathbf{x}}_1$ with $w_r = 1$. For $w_r = 3$, both $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ converged to $\tilde{\mathbf{x}}_1$.

Figure 11 show the production cost–alpha plot for

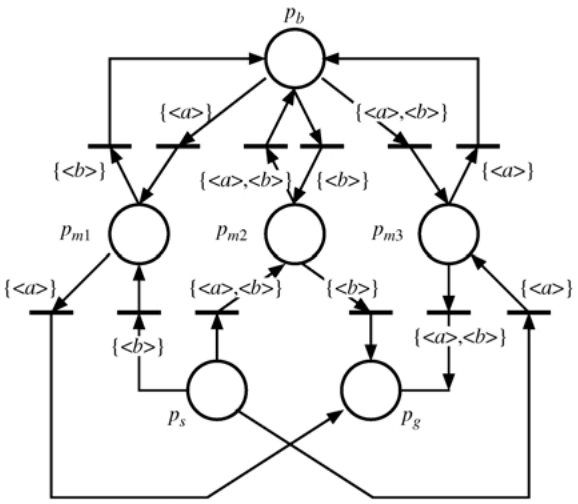

Fig. 10. Basic Petri net of the job specifications in Table 4.

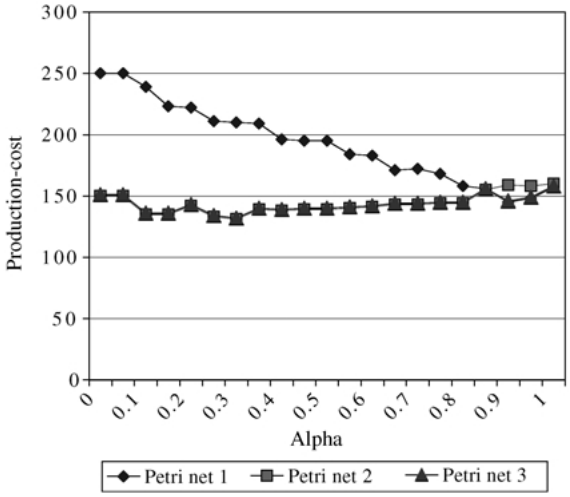

Fig. 11. Production cost–alpha plot for Example 2 with $w_r = 1$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,0,3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,0,2)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (1,0,3)$ and $\mathbf{n}(\mathbf{x}_1^*) = (2,0,2)$.

the cases for the second row of Table 5, i.e., $\mathbf{w} = (1,1,1)$. Since $\mathbf{x}_0^* = \tilde{\mathbf{x}}_0$ and $\mathbf{x}_1^* = \tilde{\mathbf{x}}_1$, the production cost of Petri net 3 over the range of $0 \le \alpha \le 1$ is exactly equal to the minimum of the ones of Petri net 1 and Petri net 2 where the ''switch'' between $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ occurs at $\alpha = 0.8$. Since Petri net 2 exhibits consistently low production cost without robustness optimization, increasing w_r further just forces Petri net 3 to be identical to Petri net 2, as seen in the bottom row of Table 5.

The robustness optimization is not at all effective in this example. Due to the extensive resource sharing in this job specifications, $\tilde{\mathbf{x}}_1$, an optimal configuration for $\alpha = 0.1$ also performs quite well for $\alpha = 0.9$. In such cases, the robustness optimizations seem not to find a configuration pair that are any better than $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$.

4.4. *Example 3*

The third example is the production scenario involving three machines and one robot conducting three operations of two jobs, as shown in Table 6.

Table 6. Job specifications for Example 3

	$J_{\langle a\rangle}$	$J_{\langle b\rangle}$
1	$M_1(10)R(3)/M_2(12)R(3)$	$M_1(7)$
2	$M_3(5)$	$M_2(5)R(2)/M_3(10)$
3	$M_1(5)R(3)/M_2(8)$	$M_3(12)$

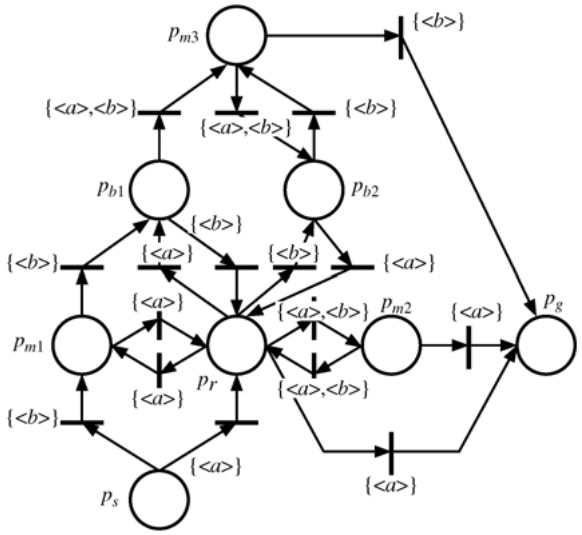

Fig. 12. Basic Petri net of the job specifications in Table 6.

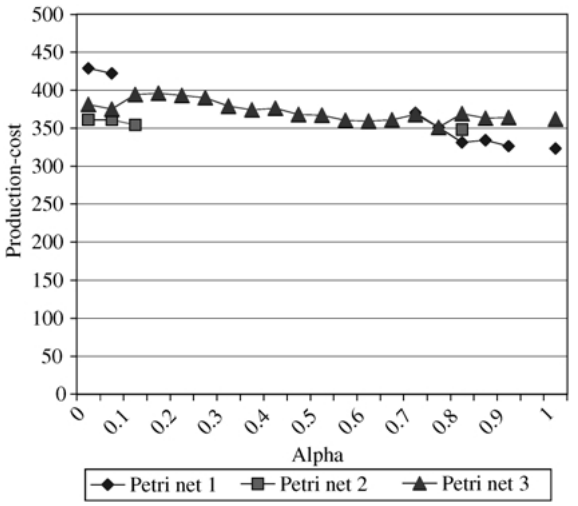

Fig. 13. Production cost–alpha plot for Example 3 with $w_r = 10$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,1,3,1)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,1,2,2)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (1,1,3,2)$ and $\mathbf{n}(\mathbf{x}_1^*) = (2,1,3,2)$.

Although all resources M_1, M_2, M_3, and R are shared between two jobs $J_{\langle a \rangle}$ and $J_{\langle b \rangle}$, some operations can only be performed by one machine type. In the job specification in Table 6, the entry of the type $M_j(t_1)R(t_2)$ means that the corresponding operations is done by the following sequence:

(1) A robot of type R carries the product to a machine of type M_j (this takes t_2).
(2) A machine of type M_j performs the operation (this takes t_1).
(3) A robot of type R takes the product away from a machine of type M_j (this takes t_2).

Figure 12 shows the basic Petri net of the job specifications in Table 6. In the figure, p_{b1} and p_{b2} are buffers (with an infinite capacity) that store both product types after the completion of the first and the second operations, respectively. The close examination of the basic Petri net reveals there is a potential of deadlock among the R, M_1, and M_2, depending on the sequence of transition firing. The total number of the machines is bounded to 10, and no more than three machines are allowed for each machine type. For the purpose of resource allocation, the robots are treated as a type of machines. In order to discourage the use of many machines, the machine costs of all types are assumed to be 20.

Table 7 shows the optimal resource allocations for different weights $\mathbf{w} = (w_m, w_f, w_r)$. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,1,3,1)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,1,2,2)$. Since they are very different from each other ($reconfig\text{-}cost(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1) = 17$), the robustness optimization cannot simply be converged to $\mathbf{x}_0^* = \tilde{\mathbf{x}}_0$ and $\mathbf{x}_1^* = \tilde{\mathbf{x}}_1$ even with $w_r = 10$, and forced to find a ''compromised'' solution, as shown in the second row of Table 7. For $w_r = 30$, both $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ converged to one configuration, which is quite different from both $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$.

Figures 13 and 14 show the production cost–alpha plot for the cases corresponding to each row in Table 5, i.e., $\mathbf{w} = (1,1,10)$ and $(1,1,30)$. The missing points in these figures indicate production cost is infinity due to the deadlock occurred during the simulation. In the case of $w_r = 10$ shown in Fig. 13, Petri net 1 and Petri net 2 experience deadlock at almost all values of α, except for the small neighborhood of the values they are optimized for. Petri net 3, on the other hand, exhibits consistently low production costs over $0 \le \alpha \le 1$, although outperformed by Petri net 1 and Petri net 2 near $\alpha = 0.9$ and $\alpha = 0.1$, respectively. Since the same SIO dispatching rule is used for all cases, this indicates

Table 7. Resource allocation result for Example 3. For comparision, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,1,3,1)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,1,2,2)$

w	$\mathbf{n}(\mathbf{x}_0^*)$	$\mathbf{n}(\mathbf{x}_1^*)$	*reconfig-cost*
$(1,1,10)$	$(1,1,3,2)$	$(2,1,3,2)$	10
$(1,1,30)$	$(3,3,3,1)$	$(3,3,3,1)$	0

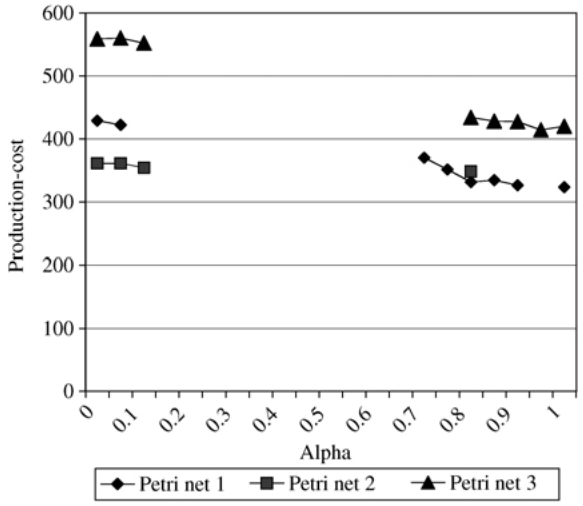

Fig. 14. Production cost–alpha plot for Example 3 with $w_r = 60$. Petri net 1 $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1, 1, 3, 1)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2, 1, 2, 2)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*)) = (3, 3, 3, 1)$ and $\mathbf{n}(\mathbf{x}_1^*) = (3, 3, 3, 1)$.

Petri net 3 avoids deadlock simply by resource allocation and by ''switching'' between $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$, which in this case occurs at $\alpha = 0.2$ and $\alpha = 0.8$. In the case of $w_r = 30$ shown in Fig. 14 however, the optimizer forces the solution to be an identical configuration. The resulting configurations are identical, but the performances became very low with deadlock occurring almost everywhere.

This example has the degree of resource sharing between Example 1 and Example 2, with an additional complexity of the potential deadlock. The robustness optimization seems to perform effectively in achieving consistently low production cost by avoiding deadlock situation. However, forcing $\mathbf{x}_0^* = \mathbf{x}_1^*$ by high w_R value degrades the quality of the solution. This is observed to some extent in Example 1, but it showed in extreme (i.e., occurrence of deadlock) in this example.

5. Concluding remarks

This paper presented a new method for designing robust flexible manufacturing systems that achieve consistently high production efficiency with minimum system reconfiguration, regardless of the changes in production plan variations within a forecasted range. The robust design of the colored Petri net model is formulated as a multi-objective optimization problem that simultaneously minimizes the production costs under multiple production plans (batch sizes for all jobs), and the reconfiguration cost due to production plan changes. A genetic algorithm, coupled with the SIO dispatching rule, is used to simultaneously find a near-optimal resource allocation and event-driven schedule of the colored Petri net model of an FMS. Although the resulting configuration is not guaranteed to be optimal, this hybrid scheme allows very fast evaluation of large number of feasible configurations, essential for the robust FMS design.

The simulation results suggest that the proposed method should be considered when the products are moderately different in their job specifications so that optimizing for a particular production plan creates inevitably bottlenecks in product flow and/or deadlock under other production plans. As a next step, we plan to investigate the classes of job specifications where this type of robustness optimization scheme is effective or non-effective. The definition of such classes would be a valuable tool for the design for manufacturing (DFM) of product families. Since a product family is typically manufactured simultaneously in a production facility, designing product families, not only for functional variety but also for manufacturing agility, would have high economical impact.

Acknowledgment

This work was carried out using computational facilities at the Design Laboratory, Department of Mechanical Engineering, the University of Michigan. This source of support is gratefully acknowledged.

References

Alla, H., Ladet, P., Martinex, J. and Silva-Suarez, M. (1985) Modeling and validation of complex systems by colored Petri nets: Application to a flexible manufacturing systems. *Lecture Notes in Computer Science*, **188**, 1–14.

Askin, R. G., Selim, H. M. and Vakharia, A. J. (1997) A methodology for designing flexible cellular manufacturing systems. *IIE Transactions*, **29**, 559–610.

Bulgak, A. A., Tarakci, Y. and Verter, V. (1999) Robust design of asynchronous flexible assembly systems. *International Journal of Production Research*, **37**(14), 3169–3184.

Chiu, Y.-F. and Fu, L.-C. (1997) A GA embedded dynamic search algorithm over a Petri net model for an fms scheduling, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 513–518.

Choi, R. and Malstrom, M. (1988) Evaluation of traditional work scheduling rules in a flexible manufacturing system with a physical simulator. *Journal of Manufacturing Systems*, **7**(1), 27–45.

David, R. and Alla, H. (1992) *Petri Net and Grafcet: Tools for Modeling Discrete Event Systems*, Prentice Hall.

Dubois, D. and Stecke, E. (1983) Using petri nets to represent production processes, in *Proceedings of the 22nd IEEE Conference on Decision and Control*, San Antonio, TX, pp. 1062–1067.

Garey, M. R. and Johnson, D. S. (1979) *Computer and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, USA.

Jafari, M. A. (1992) An architecture for a shop-floor controller using colored Petri nets. *Journal of Manufacturing Systems*, **4**(4), 159–181.

Lee, D. and DiCesare, F. (1994) Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Transaction on Robotics and Automation*, **10**, 123–132.

Lee, D. Y. and DiCesare, F. (1993) Scheduling flexible manufacturing systems with the consideration of setup times, in *Proceedings of the 1993 IEEE Conference on Decision and Control*, San Antonio, Texas, pp. 3264–3269.

Narahari, Y. and Viswanadham, N. (1985) A Petri net approach to the modeling and analysis of flexible manufacturing systems. *Annals of Operations Research*, **3**, 449–472.

Petri, C. (1962) Kommunikation mit Automaten. PhD thesis, Universitat Bonn, Bonn, West Germany.

Saitou, K. and Malpathak, S. (1999) Robustness optimization of fms under production plan variations: the case of cyclic production, in *Proceedings of the 1999 ASME Computers in Engineering Conference*, Las Vegas, Nevada, September, pp. DETC99/CIE-9127.

Saitou, K. and Qvam, H. (1988) Robustness optimization of FMS under production plan variations: preliminary results, in *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, Atlanta, Georgia, September, pp. DETC98/CIE-5691.

Shang, J. S. (1995) Robust design and optimization of material handling in an fms. *International Journal Production Research*, **33**(9), 2437–2454.

Shih, H. and Sekiguchi, T. (1991) A timed petri net and beam search based on-line fms scheduling system with routing flexibility, in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pp. 2548–2553.

Sprecher, A. (1994) *Resource-Constrained Project Scheduling*, Springer-Verlag.

Stecke, K. E. and Raman, N. (1996) Production planning decisions in flexible manufacturing systems with random material flows. *IIE Transactions*, **26**(5), 2–17.

Wu, N. Q. (1999) Necessary and sufficient conditions for deadlock-free operation in flexible manufacturing systems using a colored Petri net model. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, **29**(2), 182–204.

Zhou, M.-C. and DiCesare, F. (1996) Petri net modeling of buffers in automated manufacturing systems. *IEEE Transactions on Systems, Man, Cybernetics, Part B: Cybernetics*, **26**(1), 157–164.