Proceedings of DETC99
1999 ASME Design Engineering Technical Conferences
September 12-15, 1999, Las Vegas, Nevada, USA

DETC99/CIE-9127

ROBUSTNESS OPTIMIZATION OF FMS UNDER PRODUCTION PLAN VARIATIONS: THE CASE OF CYCLIC PRODUCTION

Kazuhiro Saitou **Samir Malpathak**
Department of Mechanical Engineering and Applied Mechanics
University of Michigan
Ann Arbor, Michigan 48109-2125
Email: {kazu,smalpath}@engin.umich.edu

ABSTRACT

This paper discusses an extension of our previous work (Saitou, 1998) on robustness optimization of flexible manufacturing systems (FMS) that undergo forecasted production plan variations. The extension is made to a more general class of FMS performing "non-linear" or "cyclic" production that allows multiple operation types per one machine type. As in our previous work (Saitou, 1998), a configuration of an FMS is modeled as a colored Petri net and the associated transition firing sequence, and the robustness of FMS is defined as the insensitivity of production performances against variations in production plan. The optimization of the robustness of the colored Petri net model is formulated as a multi-objective optimization problem which minimizes production costs under multiple production plans (batch sizes for all jobs), and reconfiguration cost due to production plan changes. A genetic algorithm, coupled with a dispatching rule based on shortest imminent operation time (SIO), is used to simultaneously find an semi-optimal resource allocation and 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 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.

INTRODUCTION

Flexible manufacturing systems (FMS) are a class of manufacturing system which can be quickly configured to produce multiple types of products (jobs). Recent increase in the use of FMS is driven by the need of agile manufacturing that can quickly adopt changes in production plans (batch sizes for all jobs) due to market demand fluctuation. While the increased flexibility of an FMS provides greater productivity under various production scenario, it imposes increased complexity in allocation of given resources to different operations required in making each product, and the scheduling of the sequence of activities to accomplish the best production efficiency (Lee, 1994).

In order to quickly adapt fluctuating market demand, the resource allocation and scheduling, or configuration in short, 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 (*i.e.*, reallocation and rescheduling) while achieving consistently efficient production under all production plans of interest (Saitou, 1998). For this, a reliable forecast on the future change in production plan must be provided, which may or may not be available at a given time.

Assuming such forecasts are available, let us consider the scenario where an FMS simultaneously produces two kinds of products A and B, and the total number of production (sum of the numbers of A's and B's to be produced) per unit time (*eg.* a day) is kept constant with production plan variation (*i.e.*, only a *fraction* of the two products varies). 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 change (imagine the extreme of this case where A and B are identical). On the other hand, when products under simultaneous production are *moderately* different, slight change in the production plan will heavily impact production efficiency, possibly due to the creation of bottlenecks in product flow. This would necessitate the system reconfiguration in order to achieve efficient production under the new production plan.

The above conjecture motivated our previous work on Petri-net based robustness optimization of FMS under production plan variation (Saitou, 1998). The simple production scenarios discussed in the work validated this conjecture for a class of FMS performing "linear" production that only allows one operation type per one machine type. This paper presents an extension of this previous work to a more general class of FMS performing "non-linear" or "cyclic" production that allows multiple operation types per one machine type. As in our previous work (Saitou, 1998), a configuration of an FMS is modeled as a colored Petri net and the associated transition firing sequence, and the robustness of FMS is defined as the insensitivity of production performances against variations in production plan. The optimization of the robustness of the colored Petri net model is formulated as a multi-objective optimization problem which minimizes production costs under multiple production plans (batch sizes for all jobs), and reconfiguration cost due to production plan changes. A genetic algorithm, coupled with a dispatching rule based on shortest imminent operation time (SIO), is used to simultaneously find an semi-optimal resource (machine) allocation and 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.

RELATED WORK

Petri nets (Petri, 1962) have been widely used for analysis and simulation of FMS due to their capability of modeling concurrency, synchronization and sequencing in discrete-event systems (Dubois, 1983; Narahari, 1985). In addition to such 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 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, 1979), an optimal schedule is often found via heuristic search algorithms such as beam search (Shih, 1991), A* algorithm (Lee, 1994) and genetic algorithms (Chiu, 1997), coupled with 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 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, 1979). GRCPSP is typically formulated as discrete programming problems and solved by heuristic search algorithms (Sprecher, 1994). The solution provides an 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, 1994), discrete-event based models such as Petri nets are rarely used for GRCPSP due to the computational time for the model simulation.

In the above work, the search is directed towards the discovery of the schedule (and the resource allocation in the case of GRCPSP) optimized for a *fixed* production plan, which could potentially be sensitive to a small perturbation in the current production plan. In continuous mathematical programming, this issue is addressed as sensitivity analyses, where the sensitivity of the optimum to small parameter perturbation is computed, in most cases, in terms of Lagrange multipliers. Several method has been proposed to find an optimal (or suboptimal) solution of nonlinear programming problems which is less sensitive to parameter perturbations (d'Entremont, 1988; Parkinson, 1990; Sundaresan, 1993). Since these methods are essentially an application of Taguchi's robust parameter design (Taguchi, 1978; Taguchi, 1987) to nonlinear programming, they are designed for continuous optimization problems, and hence do not directly apply to problems involving discrete design parameters, such as the FMS scheduling problems using Petri nets discussed above.

PROBLEM FORMULATION

Colored Petri net model of manufacturing systems

Colored Petri nets (Alla, 1985; David, 1992) are an extension of ordinary Petri nets where a place can contain multiple tokens distinguished by a "color" associated with each token. This extension allows colored Petri nets to model manufacturing systems capable of simultaneous production of multiple products in a graphically elegant manner by associating types of products with colors of tokens. As an ordinary Petri net, a colored Petri net is a directed graph consisting of two types of node, *places* and *transitions*. Two nodes are connected by an directed edge which connects either a place to a transition or a transition to a place (see Figures 1–3).

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

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

where P is a set of places, T is a set of transitions and C is a set of colors. *pre* and *post* are functions of the 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 $p \in P$ is called *marking* of the place denoted as $m(p)$, where $m : P \mapsto \mathbf{Z}^{|C|}$, and represented graphically as colored dots in a circle[1]. Let places p, q and a transition t are 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. Marking of places change according to the following rules:

1. For each input place p of a transition t, if $m(p) \geq pre(p,t,c)$ 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 definition, capacities to places 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[2], 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 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 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 <*a*> and <*b*> which both need just one operation to finish. The machines of type M_1 is capable of performing this operation on both product types <*a*> and <*b*> with an unit time, while the machines of type M_2 can only perform the operation on product <*b*> with an unit time. This job specification is summarized in Table 1, where columns indicate jobs (product types) and rows indicate operations (only one for both jobs in this example). The numbers in 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, $P = \{p_s, p_{m1}, p_{m2}, p_g\}$, $T = \{t_1, t_2, t_3, t_4\}$, $C = \{<a>, <b>\}$, $m_0(p_s) = (1,2)$, 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 FMS, the functions *pre* and *post* are simply expressed in terms of "shorthand" functions $f, g : C \mapsto C$ defined in Appendix . The colors listed next to each transition in Figures 1–3 indicates *enabling colors* of the transition, with respect of which the transition can be enabled if appears in the input place.

Table 1. example job specifications.

	$J_{<a>}$	$J_{<b>}$
1	$M_1(1)$	$M_1(1)/M_2(1)$

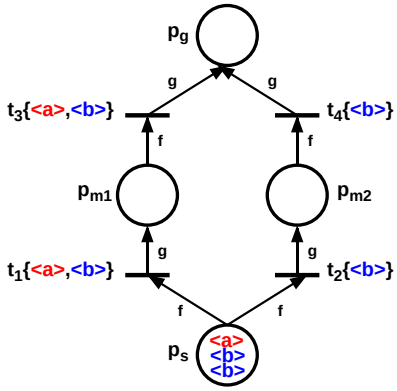

Figure 1. colored Petri net which models a production facility with start buffer p_s, two machines p_{m1} and p_{m2}, and goal buffer p_g with initial markings at clock = 0.

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

[1] In most literature, however, a token is represented as $< c >$, where c is a symbol representing the color of the token, as they are not normally printed in color.

[2] This assumes a "clock" keeping track of the marking changes.

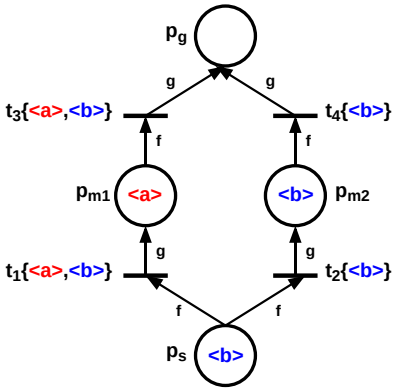

Figure 2. state of the colored Petri net after firing of t_2 with <b> at clock = 1, and t_1 with <a> at clock = 2.

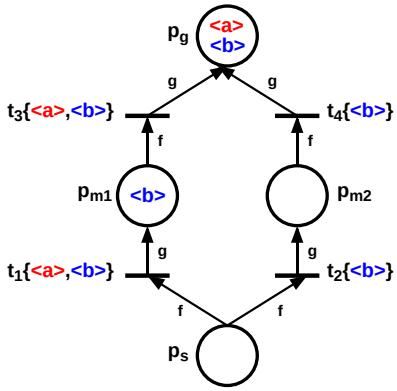

Figure 3. state of the colored Petri net after firing of t_2 with <b> at clock = 1, t_1 with <a> at clock = 2, t_3 with <a> at clock = 3, t_1 with <b> at clock = 4, and t_4 with <b> at clock = 5.

(clock = 2). The transition t_1 has the choice of firing either <a> or <b>. Suppose t_1 fires <a>. Proceeding similar to the previous firing, the token <a> is removed from p_s and appears in p_{m1} (Figure 2). This represents the system state of the machine p_{m1} processing the product <a> and the machine p_{m2} processing the product <b>[3].

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 <a> in p_{m1} to p_g. This means the machine p_{m2} has now finished processing the product <a> and send 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 <b> at clock = 4 followed by the firing of t_4 with <b> at clock = 5 would move <b> in p_g to p_{m1}, and <b> in p_{m2} to p_g (Figure 3). This represents that the machine p_{m1} is now processing the product <b> while machine p_{m2} has finished processing the product <b> and send it to the goal buffer.

At the next clock cycle (clock = 6), only t_1 can fire, which would move <b> 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 even-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.*, different schedule, that would yield a different system behavior. In general, 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]In fact, p_{m2} has finished processing <b> at this point, but the completed <b> has not yet been transfered to p_g.

Robustness optimization of FMS configurations

We consider a scenario where an FMS simultaneously produces multiple types of products which share common resources. Within this scenario, we define the *robustness of FMS* as the insensitivity of production performances against variations in production plan, and the *robustness optimization of FMS*[4] as the optimization of the robustness of FMS as defined above.

In the following, it is assumed that the *production plan* of the FMS is given in terms of the batch sizes of all jobs, *i.e.*, the numbers of each types 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 to be produced) per unit time is kept constant to, say 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 $i = 1, 2, ..., n$ 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 the forecasts on production plan changes within the timeframe of interest are 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 reconfiguration while achieving consistently efficient production under all of m production plans foreclosed*. Let $\mathbf{x}_0$ be the current configuration of the FMS, and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ be the

[4]The term "robustness optimization" should be distinguished from the similar term "robust optimization," which usually refers to the optimization algorithms whose performances are insensitive to variations in parameters of the algorithms.

future configurations corresponding the m production plans forecasted. Then, the problem can be formulated as the simultaneous minimization of the following $2m+2$ functions:

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

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

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

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

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

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

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 the configuration $\mathbf{x}$.

Reconfiguration cost from one configuration to the other 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 which need to be changed due to the change in the resource allocation. The reconfiguration cost associated with the change in schedules is not considered here since, as discussed in the following sections, dynamic scheduling with dispatching rules adopted in this work achieves the schedule change with virtually no expenses. Namely:

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

For instance, the reconfiguration cost from the colored Petri nets in Figure 4 to the one in Figure 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 <*b*>), and the addition of a second machine q_{m1} of the type M_1 (which can process <*a*> and <*b*>).

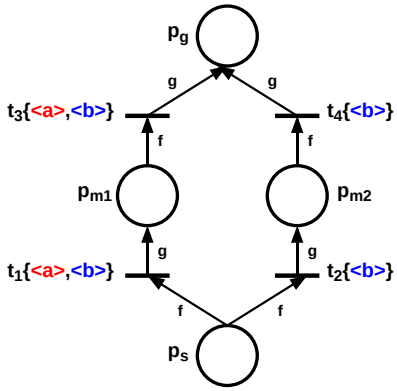

Figure 4. colored Petri net before reconfiguration.

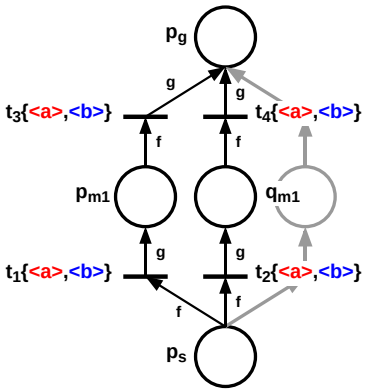

Figure 5. colored Petri net after reconfiguration.

Optimization using a genetic algorithm and dispatching rules

The robustness optimization of FMS configurations discussed in the previous section requires 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 of a FMS. Although the resulting configuration may not be guaranteed to be optimal, this hybrid scheme allows *very* fast evaluation of large number of feasible configurations.

Genetic algorithm (GA) is an optimization technique in which points in design space are analogous to organisms subject to a process of natural selection, or "survival of the fittest (Holland, 1975; Goldberg. 1989). GAs model reproduction in population of encoded representation of points (typically strings of bits) in design space – called genetic "chromosomes" – over generations. In a given generation, the quality of a chromosome,

i.e., a bit string representation of a point in design space, is measured based in 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 which 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 local rules which specifies priorities in the dispatching of products to machines while production is in progress. Dispatching rules has been traditionally used for scheduling, due to its simplicity and reliability. A number of dispatching rules such as FIFO (First-In-First-Out), SIO (Shortest Imminent Operation time) and SRPT (Shortest Remaining Processing Time) have been successfully applied to FMS scheduling (Choi, 1988). Although the schedules created by off-line heuristic 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 tends to be *robust* against the sudden change in resource allocation (*e.g.* machine breakdown), since the schedules are dynamically created during the operation, rather than determined off-line. In this work, an SIO rule is used for dynamic scheduling of a colored Petri net, whose resource allocation (the numbers of machines of each type in the job specifications) is specified by a "chromosome," of a genetic algorithm.

SIMULATION RESULTS

This section describes case studies of the robustness optimization of FMS as described in the previous section applied to the example production scenario 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.

Assumptions

We have greatly relaxed the assumptions on machine flexibility in our previous work (Saitou, 1998), so that a type of machine can possibly perform multiple manufacturing operations for multiple product types. This would allow "non-linear" or "cyclic" routing in the colored Petri net models. Accordingly, 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 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 $n = 2$, the faction vector $\mathbf{a}$ has dimension 2, and hence can be expressed using one parameter α as $(\alpha, 1-\alpha)$, where $0 \leq \alpha \leq 1$. In all examples below, the total number of production *N* per one production cycle is set to 20, and the current production plans is $\alpha = 0.9$ (*i.e.*, 18 A's and 2 B's), and the forecasted production plan is $\alpha = 0.1$ (*i.e.*, 2 A's and 18 B's).

The $2m+2$ functions as defined in Equations 2–4 with $m = 1$ is aggregated as a weighted sum:

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

where $\mathbf{x}_0$ and $\mathbf{x}_1$ are the configurations for the current production plan $\rho(\alpha_0)$ and the forecasted production plan $\rho(\alpha_1)$, and w_m, w_f and w_r are the weights of makespan, facility cost and 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 production cost of each configuration, evaluated with α varying between $\alpha = 0$ (only *A* produced) and $\alpha = 1$ (only *B* produced):

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

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

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

We shall refer to this plot as the *production cost–alpha plot*. In the production cost–alpha plot shown in the following examples, the production cost corresponding to the optimal configuration pair $(\mathbf{x}_0^*, \mathbf{x}_1^*)$ is denoted as Petri net 3, and the production cost corresponding to $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$ are denoted as called 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 smaller 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 the GALib from the MIT CADLAB with various in-house enhancements is used as an optimizer. Optimization runs

Table 2. job specifications for Example 1.

	$J_{<a>}$	$J_{<b>}$
1	$M_2(9)/M_3(3)$	$M_1(2)/M_2(6)$
2	$M_3(4)$	$M_2(7)$

took at most five minutes with a 300 MHz Sun UltraSPARC 10 Workstation.

Example 1

The first example is on the production scenario where both jobs $J_{<a>}$ and $J_{<b>}$ require two operations, and there are three machine types M_1, M_2, and M_3 available, according to the job specifications 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 in this example that the total number of the machines is bounded to 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 bound constraints. In addition, the costs of all machine types are assumed to be one.

Table 3 show 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 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 (*reconfig-cost*$(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1) = 16$), the robustness optimization cannot simply converged 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 a "compromised" solutions, as shown in the second row in Table 3. It is also observed in Table 3 that $\mathbf{n}(\mathbf{x}_0^*)$ and $\mathbf{n}(\mathbf{x}_1^*)$ becomes closer as w_r increases. For $w_r = 5$ (bottom row in Table 3), in fact, $\mathbf{n}(\mathbf{x}_0^*)$ and $\mathbf{n}(\mathbf{x}_1^*)$ converged to an identical value (*reconfig-cost*=0), which is quite different from both $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_1$.

Figures 7, 8 and 9 show the production cost–alpha plot for the cases corresponding to each rows in 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 Figurers 7 and 8, respectively. Since for $w_r = 1$ the optimizer is not strongly forced to find $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ that are close to each other, $\mathbf{x}_0^*$ tends to

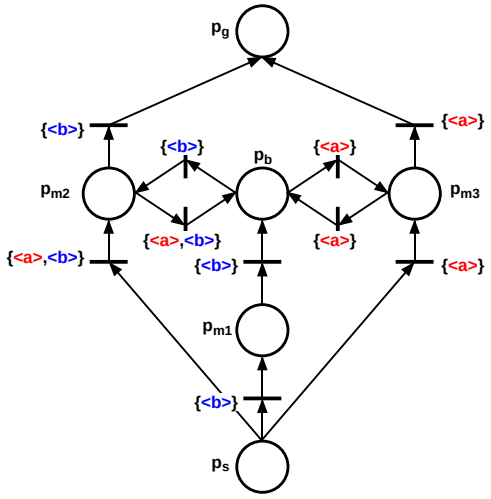

Figure 6. basic Petri net of the job specifications in Table 2. During the optimization process, the Petri net corresponding to a particular resource allocation is constructed by adding or removing machines of each type in this basic Petri net.

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

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 $0 \le \alpha \le 1$ being quite similar to the *minimum* of the ones of Petri net 1 and Petri net 2, as shown in Figures 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 $w_r = 3$ (Figure 8) and $w_r = 5$ (Figure 9). Although converged to one configuration, 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 $0 \le \alpha \le 0.5$. In other words, $\mathbf{x}_0^* = \mathbf{x}_1^*$ is achieved with the price of higher production costs.

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 given job specifications cause $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_0$ quite different. Since they are quite different, production flow bottlenecks are quickly created when running $\tilde{\mathbf{x}}_0$ with $\alpha < 0.9$ or

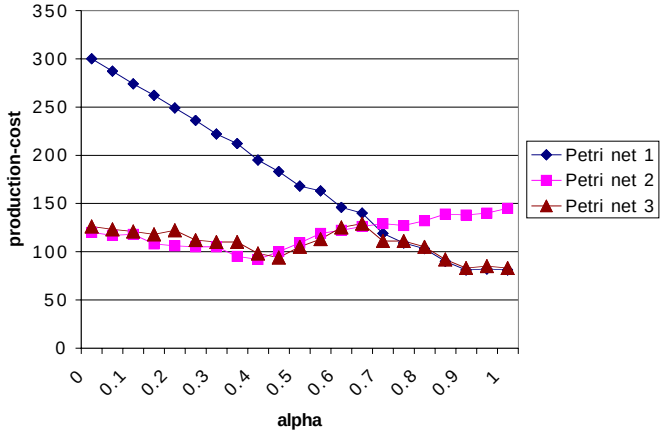

Figure 7. production cost–alpha plot for Example 1 with $w_r = 1$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0,1,3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0,3,1)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (0,1,3)$ and $\mathbf{n}(\mathbf{x}_1^*) = (1,2,1)$.

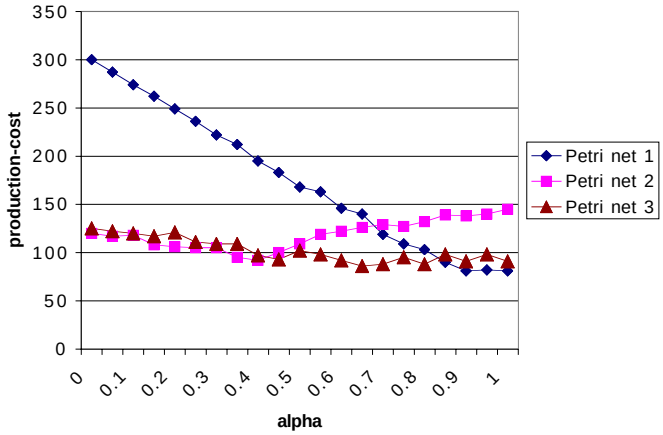

Figure 8. production cost–alpha plot for Example 1 with $w_r = 3$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0,1,3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0,3,1)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (0,2,2)$ and $\mathbf{n}(\mathbf{x}_1^*) = (1,2,1)$.

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 $J_{<b>}$, which is quite slow. In such cases, the robustness optimization seems effectively find a configuration pair that exhibit the robust performances over a range of α.

Example 2

The second example is taken from Example 2 of (Lee, 1993). This example is similar to Example 1 with more extensive re-

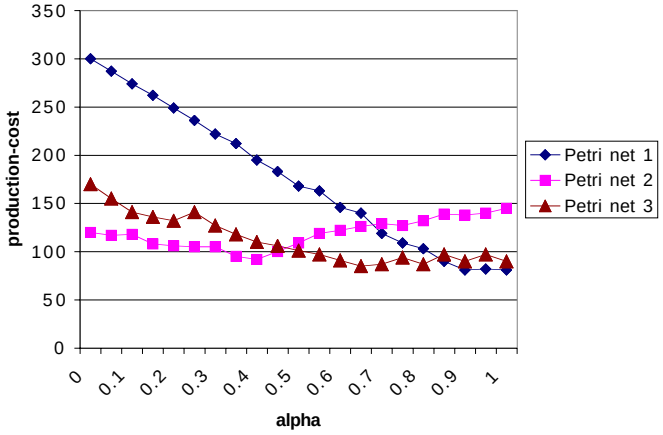

Figure 9. production cost–alpha plot for Example 1 with $w_r = 5$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0,1,3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0,3,1)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (0,2,2)$ and $\mathbf{n}(\mathbf{x}_1^*) = (0,2,2)$.

Table 4. job specifications for Example 1.

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

source sharing among jobs. The job specification shown in Table 4 indicates that all machine types M_1, M_2 and M_3 are shared between two jobs $J_{<a>}$ and $J_{<b>}$. This extensive resource sharing creates complex routings among machines and a buffer, as illustrated in the basic Petri net shown in Figure 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 optimal resource allocations for different weights $\mathbf{w} = (w_m, w_f, w_r)$. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (0,1,3)$ and $\mathbf{n}(\tilde{\mathbf{x}}_1) = (0,3,1)$. Since they are fairly close, (*reconfig-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 the cases corresponding to the second row in 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 $0 \leq \alpha \leq 1$ is exactly 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

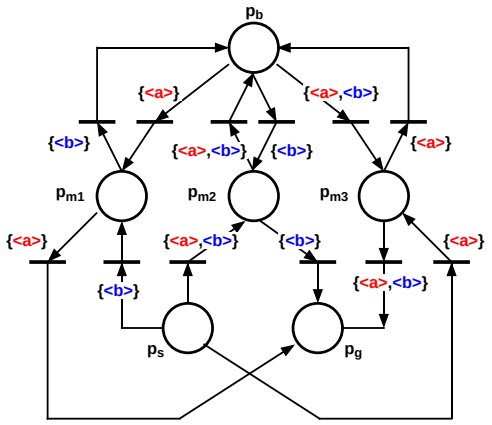

Figure 10. basic Petri net of the job specifications in Table 4.

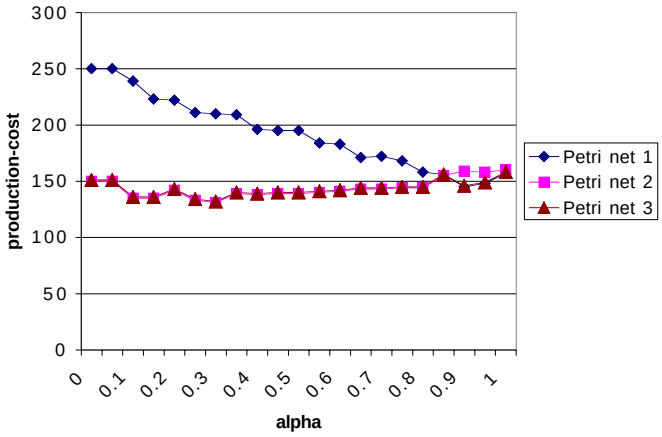

Figure 11. production cost–alpha plot for Example 2 with $w_r = 1$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,0,3)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,0,2)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (1,0,3)$ and $\mathbf{n}(\mathbf{x}_1^*) = (2,0,2)$.

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

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 seems not to find a configuration pair which are any better than $\tilde{\mathbf{x}}_0$ or $\tilde{\mathbf{x}}_1$.

Example 3

The third example is the production scenario involving three machines and one robot that conduct three operations of two jobs, as shown in Table 6. Although all all resources M_1, M_2, M_3, and R are shared between two jobs $J_{<a>}$ and $J_{<b>}$, some operations can only be performed by one machine type. In the job specification 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).

Table 6. job specifications for Example 3.

	$J_{<a>}$	$J_{<b>}$
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	$M_1(5)R(3)/M_2(8)$	$M_3(12)$

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. Close examination of the basic Petri net reveals that there are potential of deadlock situation among the R, M_1 and M_2, depending on the sequence of transition firing. The total number of the machines in this example is bounded to ten, and no more than three machines are allowed for each machine type[5]. In order to discourage the use of many machines, the costs of all machine types are assumed to be twenty.

Table 7 shows 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 each other (*reconfig-cost*$(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1) = 17$), the robustness optimization cannot simply 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" solutions, as shown in the second row in 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 $\tilde{\mathbf{x}}_1$.

[5]For the purpose of resource allocation, the robots are treated as a type of machines.

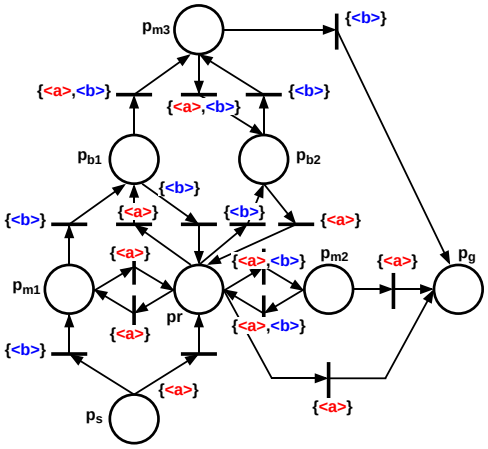

Figure 12. basic Petri net of the job specifications in Table 6.

Table 7. resource allocation result for Example 3. For comparison, $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,1,3,1)$ 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

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 deadlock occurred during the simulation. In the case of $w_r = 10$ shown in Figures 13, Petri net 1 and Petri net 2 experience deadlock at almost any 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 $0 \leq \alpha \leq 1$, although outperformed by Petri net 1 and Petri net 2 near $\alpha = 0.9$ and $\alpha = 0.1$, respectively. Since the same SIO dispatching rules are used for all cases, this indicates Petri net 3 avoids deadlock simply by resource allocation and "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 Figures 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 degree of resource sharing between Example 1 and Example 2, with additional complexity of the potential deadlock. The robustness optimization seems to perform effectively in achieving the 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.

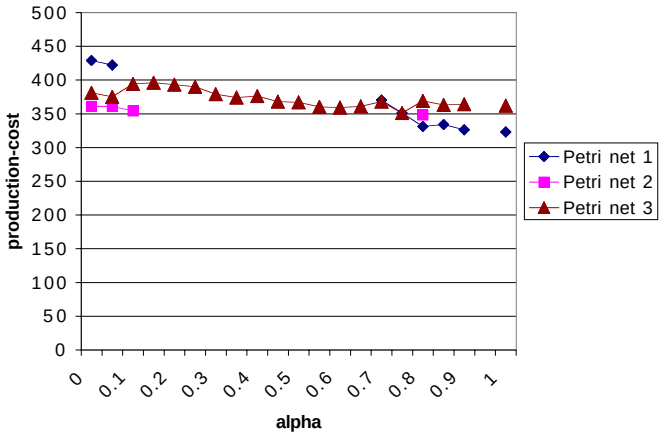

Figure 13. production cost–alpha plot for Example 3 with $w_r = 10$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,1,3,1)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,1,2,2)$), Petri net 3: $\mathbf{n}(\mathbf{x}_0^*) = (1,1,3,2)$) and $\mathbf{n}(\mathbf{x}_1^*) = (2,1,3,2)$).

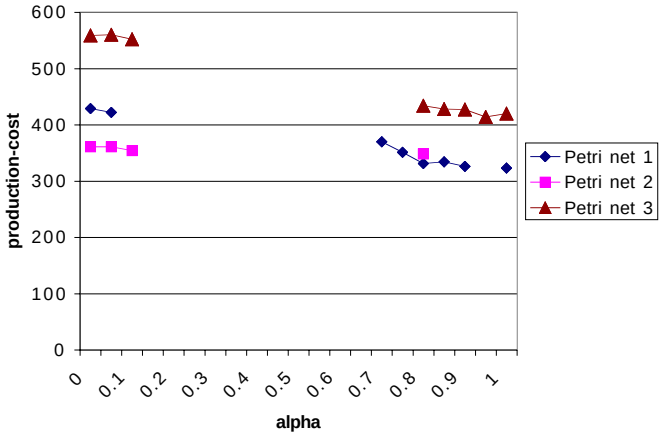

Figure 14. production cost–alpha plot for Example 3 with $w_r = 60$. Petri net 1: $\mathbf{n}(\tilde{\mathbf{x}}_0) = (1,1,3,1)$, Petri net 2: $\mathbf{n}(\tilde{\mathbf{x}}_1) = (2,1,2,2)$, Petri net 3: $\mathbf{n}(\mathbf{x}_0^*)) = (3,3,3,1)$ and $\mathbf{n}(\mathbf{x}_1^*) = (3,3,3,1)$.

DISCUSSION AND FUTURE WORK

This paper presented an extension of our previous work (Saitou, 1998) on robustness optimization of flexible manufacturing systems (FMS) that undergo forecasted product plan variations. The extension is made to a more general class of FMS performing "non-linear" or "cyclic" production that allows multiple operation types per one machine type. As in our pre-

vious work (Saitou, 1998), a configuration of an FMS is modeled as a colored Petri net and the associated transition firing sequence, and the robustness of FMS is defined as the insensitivity of production performances against variations in production plan. The optimization of the robustness of the colored Petri net model is formulated as a multi-objective optimization problem which minimizes production costs under multiple production plans (batch sizes for all jobs), and reconfiguration cost due to production plan changes. A genetic algorithm, coupled with a dispatching rule based on shortest imminent operation time (SIO), is used to simultaneously find an semi-optimal resource allocation and 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 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. As a next step, we plant to investigate the classes of job specifications to which this type of robustness optimization scheme is effective or non-effective. Definition of such classes would be an very useful tool for design for manufacturing (DFM) of products families. Since they are naturally manufactured in the situation similar to the one considered in this paper, 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 Computational Design Laboratory, Department of Mechanical Engineering and Applied Mechanics, the University of Michigan. This source of support is gratefully acknowledged.

Appendix A

The *pre* and *post* functions of the example colored Petri net in Figures 1–3 are defined as follows:

$$
\begin{aligned}
pre(p_s, t_1, \langle a \rangle) &= f(\langle a \rangle) = \langle a \rangle = (1,0) \\
pre(p_s, t_1, \langle b \rangle) &= f(\langle b \rangle) = \langle b \rangle = (0,1) \\
pre(p_s, t_2, \langle b \rangle) &= f(\langle b \rangle) = \langle b \rangle = (0,1) \\
pre(p_{m1}, t_3, \langle a \rangle) &= f(\langle a \rangle) = \langle a \rangle = (1,0) \\
pre(p_{m1}, t_3, \langle b \rangle) &= f(\langle b \rangle) = \langle b \rangle = (0,1) \\
pre(p_{m2}, t_4, \langle b \rangle) &= f(\langle b \rangle) = \langle b \rangle = (0,1) \\
post(p_{m1}, t_1, \langle a \rangle) &= g(\langle a \rangle) = \langle a \rangle = (1,0) \\
post(p_{m1}, t_1, \langle b \rangle) &= g(\langle b \rangle) = \langle b \rangle = (0,1) \\
post(p_{m2}, t_2, \langle b \rangle) &= g(\langle b \rangle) = \langle b \rangle = (0,1) \\
post(p_g, t_3, \langle a \rangle) &= g(\langle a \rangle) = \langle a \rangle = (1,0) \\
post(p_g, t_3, \langle b \rangle) &= g(\langle b \rangle) = \langle b \rangle = (0,1) \\
post(p_g, t_4, \langle b \rangle) &= g(\langle b \rangle) = \langle b \rangle = (0,1)
\end{aligned}
$$

For other points (p,t,c) not defined above, $pre(p,t,c)$ and $post(p,t,c)$ are undefined.

REFERENCES

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

Y.-F. Chiu and L.-C. Fu. 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*, pages 513–518, 1997.

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

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

K. d'Entremont and K. Ragsdell. Design for latitude using TOPT. In *ASME Advances in Design Automation*, volume DE-Vol. 14, pages 265–272, 1988.

A. Dhumal, R. Dhawan, A. Kona., and A. Soni. Reconfigurable system analysis for agile manufacturing. In *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine, California, August 18–22 1996.

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

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

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

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

D. Y. Lee and F. DiCesare. Scheduling flexible manufacturing systems with the consideration of setup times. In *Proceedings of the 1993 IEEE Conference on Desicion and Control*, pages 3264–3269, San Antonio, Texas, 1993.

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

Y. Narahari. and N. Yiswanadham. A petri net approach to the modeling and analysis of flexible manufacturing systems. *Annals of Operations Research*, 3:449–472, 1985.

A. Parkinson, C. Sorensen, J. Free, and B. Canfield. A petri net approach to the modeling and analysis of flexible manufacturing systems. In *Proceedings of the 1990 ASME Design Automation Conference*, volume 2, pages 121–128, Chicago, Illinois, September 1990.

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

K. Saitou and H. Qvam. Robustness optimization of FMS under production plan variations: preliminary results. In *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, number DETC98/CIE-5691, September 1998.

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

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

S. Sundaresan, K. Ishii, and D. Houser. A robustness optimization procedure with variations on design variables and constraints. In *ASME Advances in Design Automation*, volume DE-Vol. 65-1, pages 379–386, 1993.

G. Taguchi. Off-line and on-line quality control systems. In *Proceedings of the International Conference on Quality Control*, Tokyo, Japan, 1978.

G. Taguchi. Systems of experimental design. In Don Clausing, editor, *American Supplier Institute*, Deaborn, Michigan, 1987.