CONTROL POLICIES CONCILIATING DEADLOCK AVOIDANCE AND FLEXIBILITY IN FMS RESOURCE ALLOCATION

M.P. Fanti, B. Maione, S. Mascolo, B. Turchiano

Dipartimento di Elettrotecnica ed Elettronica - Politecnico di Bari
Via Re David 200 - 70125 BARI , ITALY
E-mail : maione@poliba.it

Abstract

Deadlock arises in FMS when a job set is in "circular wait", i.e. each job in the set waits for a resource held by another job in the same set. This is an unfavourable situation that can occur in production systems with high routing flexibility. Facing this problem requires control policies that avoid deadlock conditions by ruling resource allocation and deallocation. Often, by reducing the allowed options in resource usage, these policies lead to poor production system performance. On the other hand, more flexible approaches to deadlock avoidance usually need more complex control algorithms.

This paper considers two deadlock avoidance policies that use feedback of the production system state to allow or inhibit resource allocation or deallocation. Both the policies are stated using a digraph-theoretic approach describing possible and current job-resource interactions. The former policy has been already introduced by the authors in a previous work. The latter is original and, by extending the basic ideas that motivate the previous one, allows wider flexibility in resource use, even if it involves higher computational complexity. The paper compares the two policies and a third one proposed by other authors, by analysing two examples by discrete event simulation. This confirms that the new policy results in better performance measures.

1 Introduction

Deadlock can arise in Flexible Manufacturing Systems (FMS) with high routing flexibility [11, 12]. It occurs when each element from a set of jobs (pieces, parts) waits for a resource (machine, buffer slot, transport unit, etc.) held by another job in the same set. All the pieces and the resources involved in such a condition remain blocked indefinitely unless a suitable recovery policy restores regular production flow. In the last years several researchers have dealt with this subject [1, 5, 6, 8, 9, 11, 12], introducing avoidance policies that use feedback control laws to rule resource allocation and deallocation. Such policies differ from one another in computation complexity and in constraint tightness they impose on the resource utilization. In particular, an excessive reduction of the flexibility in resource allocation could lead to poor performance of the production system. On the other hand, higher flexibility usually asks for more complex deadlock avoidance policies.

This paper compares two avoidance policies allowing considerable flexibility in resource allocation: the former has been introduced in [6], while the latter is a new proposal of the authors. The policies involve different computation complexities and show different restrictions in limiting resource usage. Both of them are stated by means of graph-theoretic tools and assume the system be modelled as a Discrete Event Dynamical System (DEDS) [3, 4, 13]. The paper is organized in six Sections. The next Section introduces notations and summarizes the main theoretical results developed in [6, 7], while Section 3 describes the two deadlock avoidance policies. Section 4 discusses the computational complexity. In

0-7803-2535-4/95 $4.00

Section 5 we apply the proposed methods and a further policy introduced by Banaszak and Krogh [1] to two examples, using discrete event simulation to compare the resulting performance indices of the production. Finally, Section 6 draws the conclusions.

2 Basic Definitions and Previous Results

In an FMS $\mathcal{S}$ each part requires service from the system resources in a specific sequence, called Working Procedure. We assume that each Working Procedure terminates with a fictitious resource (r_R) representing the completion of the job processing. So, if $\mathcal{J}$ indicates the set of jobs to produce, each part $j \in \mathcal{J}$ follows a particular Working Procedure w, from a set $\mathcal{W}$. We suppose that each part holds one resource at a time, in exclusive mode. In our approach, detection and avoidance of deadlocks involve two main digraphs [2]. The former, named Working Procedure Digraph and denoted by $D_{\mathcal{W}}=(\mathcal{R},\mathcal{E}_{\mathcal{W}})$, shows the specific order in which resources appear in all the Working Procedures. To be specific, each vertex of the set $\mathcal{R}$ corresponds to a resource r_i ($i=1,\ldots,R$). An edge e_{im} is an ordered pair (r_i,r_m) of nodes in $\mathcal{R}$ such that $e_{im}=(r_i,r_m)$, directed from r_i to r_m, belongs to $\mathcal{E}_{\mathcal{W}} \subset \mathcal{R}\times\mathcal{R}$ iff (if and only if) r_m immediately follows r_i in some $w \in \mathcal{W}$.

The second main digraph, named Transition Digraph, describes the progress of jobs in process, i.e. the dynamic interaction job-resources. Assuming that the system dynamics is described by a DEDS, we denote by $q(t) \in \mathcal{Q}$ the $\mathcal{S}$-state at time t, where $\mathcal{Q}$ is the complete state set. In particular, at each time t, q must describe the set $\mathcal{J}_q$ of the jobs in process, the resources currently held by each job $j \in \mathcal{J}_q$, the Working Procedures associated with such jobs and, finally, the Residual Working Procedures, i.e. the sequence of resources necessary for each $j \in \mathcal{J}_q$ to complete the processing. The Transition Digraph $D_{Tr}(q)=[\mathcal{R},\mathcal{E}_{Tr}(q)]$ is defined as follows: edge $e_{im} \in \mathcal{E}_{Tr}(q)$ iff a job $j \in \mathcal{J}_q$ holds r_i in the state q and requires r_m as next resource. Hence $D_{Tr}(q)$ describes all the next transitions of jobs in $\mathcal{S}$. Not always a transition represented by an edge $e_{im} \in \mathcal{E}_{Tr}(q)$ (for brevity: transition e_{im}) can be immediately executed. Namely, for e_{im} to occur, r_m must be *idle* in the state q. In this case, we call e_{im} a *feasible* transition for the state q. Vice versa, if r_m is *busy*, we say that e_{im} is a *blocked* transition for q.

The Transition Digraph allows deadlock detection: namely, as shown by Fanti *et al.* [6], q is a deadlock state for $\mathcal{S}$ iff there is at least one cycle in $D_{Tr}(q)$. Such a cycle means that there is a set of jobs which remain indefinitely blocked, in circular wait. In the following we denote a deadlock condition as First Level Deadlock (FLD for brevity). Now let $\gamma_n=(\mathcal{R}_n,\mathcal{E}_n)$ be a cycle of $D_{\mathcal{W}}$. We define *Cycle Capacity* of γ_n the number of vertices of such a cycle, i.e. the integer $C(\gamma_n)=\mathrm{Card}(\mathcal{R}_n)$. Moreover the *Overlap Degree* of γ_n in the state q is the integer $O_q(\gamma_n)=\mathrm{Card}[\mathcal{E}_n \cap \mathcal{E}_{Tr}(q)]$. Obviously, a cycle γ_n of $D_{\mathcal{W}}$ is in deadlock condition in $D_{Tr}(q)$ iff $O_q(\gamma_n)=C(\gamma_n)$ and indicates the number of edges of γ_n appearing in $D_{Tr}(q)$.

To define deadlock avoidance policies it is essential to prevent the occurrence of some critical situations, named "impending system deadlocks" [9]. These are not FLD, even if they necessarily evolve to circular waits in the immediate future. One of such situations is the Second Level Deadlock (SLD) where a job set is not currently deadlocked, but it progresses inevitably to a deadlock condition at the next transition. As an example let us consider a system introduced by Wysk *et al.* [11] including three Working Procedures: $w_1=(r_1,r_2,r_4,r_5)$, $w_2=(r_3,r_1,r_4,r_5)$ and $w_3=(r_4,r_3,r_2,r_1,r_5)$, where r_5 stands for the fictitious resource. Figure 1.a gives $D_{\mathcal{W}}$, where each edge is labelled by the corresponding Working Procedure(s). Solid lines in Figure 1.b form $D_{Tr}(q)$ for a SLD state; moreover

dashed lines represent second transitions in the Residual Working Procedures and dark nodes indicate *busy* resources. It is easy to realize that executing the feasible transition e_{32} or e_{12} results in a FLD for cycle $\gamma_1=(\{r_1,r_2\},\{e_{12},e_{21}\})$ or $\gamma_2=(\{r_2,r_4,r_3\},\{e_{24},e_{43},e_{32}\})$, respectively.

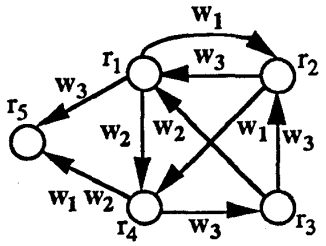


Figure 1.a: Digraph D_{Ψ} for Example 1.

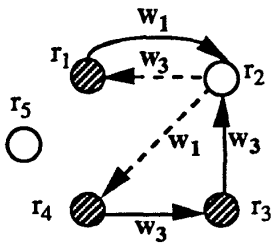


Figure 1.b: A SLD state

Now we may go one step further by introducing the cycle set Γ_g, such that $\aleph_n \cap \aleph_h = \{r_g\}$ for any $\gamma_n,\gamma_h \in \Gamma_g$, with $\gamma_n \neq \gamma_h$. The set Γ_g is called *rosace* of D_{Ψ} and the only resource shared by all the elements from Γ_g (i.e. r_g) is the centre of rosace Γ_g [2]. Denoting by $\aleph_{\Gamma_g}$ and $\mathcal{E}_{\Gamma_g}$ respectively the sets of all the vertices and edges in the cycles from Γ_g, we define the rosace capacity as $C(\Gamma_g)=Card(\aleph_{\Gamma_g})$ and the rosace overlap degree as $O_q(\Gamma_g)=Card[\mathcal{E}_{\Gamma_g} \cap \mathcal{E}_{Tr}(q)]$.

According to the proof given by the authors [6], necessary and sufficient conditions for q to be a SLD state is that D_{Ψ} must contain a rosace Γ_g with r_g *idle* and all the other resources *busy* in the state q. Moreover, for each cycle $\gamma_n \in \Gamma_g$ there exists $\gamma_h \in \Gamma_g$ such that the only *feasible* transition of γ_h determines the deadlock of γ_n. This implies that a Working Procedure $w \in \Psi$ contains vertices r_i, r_g and r_p in strict order of succession, with $e_{ig} \in \mathcal{E}_h$ and $e_{gp} \in \mathcal{E}_n$.

The previous conditions point out a connection between the topology of D_{Ψ} and the possibility of SLD occurrences. F.e. it is easy to verify that D_{Ψ} of Figure 1.a contains the rosace $\Gamma_1=\{\gamma_1,\gamma_2\}$ with centre r_2. Moreover the Working Procedure w_1 has vertices r_1, r_2 and r_4 in strict order of succession, and w_3 includes the sequence r_3, r_2 and r_1. Hence the system enjoys the conditions described above.

Starting from the concept of SLD, it is easy to introduce the notion of Third Level Deadlock (TLD). This situation characterizes states in which a job set is neither in FLD nor in SLD but it certainly progresses to a FLD or a SLD at the next transition. Figure 2.a shows the digraph D_{Ψ} for a system with the following Working Procedures: $w_1=(r_1,r_8,r_2,r_3,r_4,r_9)$, $w_2=(r_5,r_4,r_3,r_6,r_9)$, $w_3=(r_5,r_6,r_3,r_7,r_9)$ and $w_4=(r_7,r_3,r_2,r_1,r_8,r_9)$. Clearly, D_{Ψ} contains five cycles $\gamma_1=(\{r_1,r_8,r_2\},\{e_{18},e_{82},e_{21}\})$, $\gamma_2=(\{r_2,r_3\},\{e_{23},e_{32}\})$, $\gamma_3=(\{r_3,r_4\},\{e_{34},e_{43}\})$, $\gamma_4=(\{r_6,r_3\},\{e_{63},e_{36}\})$ and $\gamma_5=(\{r_3,r_7\},\{e_{37},e_{73}\})$. Figure 2.b gives the digraph $D_{Tr}(q)$ (solid lines) for a TLD state. For sake of simplicity, the digraphs of Fig. 2.a and 2.b do not include the fictitious resource r_9: namely it can not pertain to any cycle [6]. Dashed lines in Fig. 2.b indicate second and third transitions in the Residual Working Procedures. Executing e_{43}, or e_{63} leads to FLD condition for γ_4 or γ_5, respectively, while e_{73} and e_{82} determine a SLD for rosaces $\Gamma_1=\{\gamma_1,\gamma_2\}$ or $\Gamma_2=\{\gamma_2,\gamma_3,\gamma_4,\gamma_5\}$, respectively. Obviously, the notions of SLD and TLD can be generalized to define higher level deadlocks.

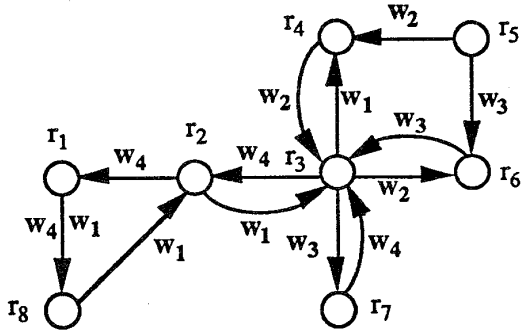


Figure 2.a: Digraph $D_{\mathcal{W}}$ for Example 2.

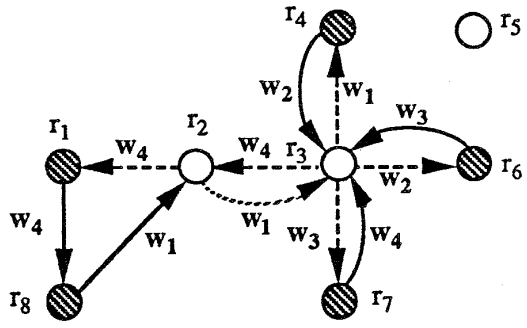


Figure 2.b: A TLD state

As proven in [7], for a TLD to occur the topology of $D_{\mathcal{W}}$ must enjoy some properties. Namely, if $\mathbf{q}$ is a TLD state then $D_{\mathcal{W}}$ contains two rosace sets ($\mathcal{H}_1$ and $\mathcal{H}_2$) each one characterized by a unique centre for all its elements (r_{g1} and r_{g2}, respectively, with $r_{g1} \neq r_{g2}$). All the rosaces in both sets share one cycle and any two rosaces (in the same set or each from a set) only share the vertices and the edges of such a cycle. Moreover all the resources of the rosaces in $\mathcal{H}_1 \cup \mathcal{H}_2$ are *busy* but r_{g1} and r_{g2} that are *idle*. Finally, for each rosace $\Gamma_g \in \mathcal{H}_1$ there exists a *feasible* transition belonging to a rosace $\Gamma_m \in \mathcal{H}_2$ that puts Γ_g in SLD, and vice versa.

From the example shown in Figure 2.a we have $\mathcal{H}_1=\{\Gamma_1\}$ and $\mathcal{H}_2=\{\Gamma_2\}$ with centre r_2 and r_3, respectively. The sets $\mathcal{H}_1$ and $\mathcal{H}_1$ share only the cycle γ_2. Moreover all the resources in $\mathcal{H}_1 \cup \mathcal{H}_2=\{\Gamma_1,\Gamma_2\}$ are *busy*, but r_2 and r_3.

Finally, let $\mathcal{H}_i$ be a rosace set of $D_{\mathcal{W}}$ and let $\mathcal{N}_{\mathcal{H}_i}$ and $\mathcal{E}_{\mathcal{H}_i}$ be respectively the sets of all the vertices and of all the edges pertaining to the elements of $\mathcal{H}_i$. Then we define the rosace set Capacity and the rosace set Overlap Degree as $C(\mathcal{H}_i)=\mathrm{Card}(\mathcal{N}_{\mathcal{H}_i})$ and $O_{\mathbf{q}}(\mathcal{H}_i)=\mathrm{Card}[\mathcal{E}_{\mathcal{H}_i} \cap \mathcal{E}_{Tr}(\mathbf{q})]$, respectively.

3 Deadlock Avoidance Techniques

Deadlock avoidance policies express feedback control laws (named *Restriction Policies*) that utilize information on the current state $\mathbf{q}$ to inhibit or to enable events modifying the resource allocation. In particular we consider events of the following two types: a new job enters the system (1-type event) and a job progresses from a resource to another one or it leaves the system (2-type event). To describe a 1-type event we use a pair (j,w), where $j \in \mathcal{J}$ is the job entering $\mathcal{S}$ and $w \in \mathcal{W}$ is the Working Procedure that it has to follow. Moreover we specify a 2-type event with the *feasible* transition e_{im} representing the job progress.

With reference to the following sets: $\mathcal{X}_1 = \{(\mathbf{q},w) \in \mathcal{Q} \times \mathcal{W}$: the first resource of w is *idle* in $\mathbf{q}\}$ and $\mathcal{X}_2 = \{(\mathbf{q},e_{im}) \in \mathcal{Q} \times \mathcal{E}_{\mathcal{W}}$: e_{im} is *feasible* in $\mathbf{q}\}$, we define a Restriction Policy as a pair of functions:

$$f_1 : \mathcal{X}_1 \rightarrow \{0,1\} \qquad f_2 : \mathcal{X}_2 \rightarrow \{0,1\}$$

where $f_1(\mathbf{q},w)=0$ ($f_1(\mathbf{q},w)=1$) means that, for $\mathcal{S}$ in the state $\mathbf{q}$, every 1-type event associated with w is inhibited (enabled) and $f_2(\mathbf{q},e_{im})=0$ ($f_2(\mathbf{q},e_{im})=1$) indicates that, for $\mathcal{S}$ in the state $\mathbf{q}$, 2-type event associated with the *feasible* transition e_{im} is inhibited

(enabled).

Avoiding deadlock means to prevent situations of any level deadlock from occurring. One possible approach to the deadlock avoidance consists in suitably limiting the number of pieces in process. For example, imposing that such a number be less than the minimum capacity of all the cycles in $D_{\mathcal{W}}$, makes impossible the occurrence of any level deadlock "a priori". However this policy is too restrictive and results in poor production performance. A more flexible approach limits the number of jobs in progress so that only a SLD and higher level deadlocks are "a priori" prevented from occurring. In addition, all those events which just lead to a FLD are inhibited. To make this idea more clear and to state the first Restriction Policy, further definitions are still necessary.

Let $\{\gamma_1,\dots \gamma_n\dots, \gamma_N\}$ be the complete set of all the cycles of $D_{\mathcal{W}}$. We define the Second Level Digraph $D^2_{\mathcal{W}}=(\mathcal{N}^2,\mathcal{E}^2_{\mathcal{W}})$, by associating a vertex n^2_n with each cycle γ_n of $D_{\mathcal{W}}$, so that $\mathcal{N}^2=\{n^2_1,\dots,n^2_n,\dots,n^2_N\}$. Moreover, the edge $e^2_{vs}=(n^2_v,n^2_s)$ belongs to $\mathcal{E}^2_{\mathcal{W}}$, iff γ_v and γ_s have only one vertex in common (say r_m) and there exists a Working Procedure $w\in\mathcal{W}$, containing vertices r_i, r_m and r_p in strict order of succession, with $e_{im}\in\mathcal{E}_v$ and $e_{mp}\in\mathcal{E}_s$. Now let $\gamma^2_n=(\mathcal{N}^2_n,\mathcal{E}^2_n)$ indicate a cycle of $D^2_{\mathcal{W}}$ (second level cycle). The Cycle Capacity, $C(\gamma^2_n)$, of the second level cycle γ^2_n equals the number of resources in the cycles from Γ_n, i.e. $C(\gamma^2_n)=C(\Gamma_n)$. Moreover, we denote by $O_q(\gamma^2_n)$ the Overlap Degree of γ^2_n in the state $\mathbf{q}$, where $O_q(\gamma^2_n)=O_q(\Gamma_n)$. A crucial role in the Restriction Policies proposed in the following is played by a particular subset of the second level cycles. It is defined as $\Gamma^2=\{\gamma^2_1,\dots,\gamma^2_n,\dots,\gamma^2_G\}$, where, for any n=1,2,...,G, the vertices in $\mathcal{N}^2_n$ correspond to a rosace Γ_n of $D_{\mathcal{W}}$.

Given such definitions, the following Proposition proven in [6] introduces a necessary condition for $\mathbf{q}$ to be a SLD state.

Proposition 1: Necessary condition for $\mathbf{q}$ to be a SLD state is that the Second Level Digraph $D^2_{\mathcal{W}}$ has a cycle $\gamma^2_n\in\Gamma^2$, whose overlap degree in the state $\mathbf{q}$ is $O_q(\gamma^2_n)=[C(\gamma^2_n)-1]$.

The previous result motivates the first deadlock free Restriction Policy.

Restriction Policy A (RPA)

$f_1(\mathbf{q},w_k)=1$ if $D_{Tr}(\mathbf{q}')$ does not contain any cycle and $Card(\mathcal{J}_{q'})<(C^2_0-1)$;
$f_1(\mathbf{q},w_k)=0$ otherwise;

$f_2(\mathbf{q},e_{im})=1$ if $D_{Tr}(\mathbf{q}')$ does not contain any cycle;
$f_2(\mathbf{q},e_{im})=0$ otherwise

where $\mathbf{q}'$ indicates the state obtained as a consequence of the considered 1-type or 2-type event and

$$C^2_0=\min_{\Gamma^2} C(\gamma^2_n)$$

is an integer, with $C^2_0=\infty$ if Γ^2 is empty.

The function f_1 bounds the number of jobs in progress so that the necessary conditions of Proposition 1 fail. Namely, since the previous definitions imply $O_{q'}(\gamma^2_n)\le Card(\mathcal{J}_{q'})$ and $[C^2_0-1]\le[C(\gamma^2_n)-1]$, the function f_1 leads to $O_{q'}(\gamma^2_n)\le[C(\gamma^2_n)-1]$ for any $\gamma^2_n\in\Gamma^2$. Even if this condition avoids any SLD, it does not keep FLD from occurring. Thus, to inhibit events just leading to a FLD, f_1 and f_2 impose that $D_{Tr}(\mathbf{q}')$ is acyclic.

To further improve flexibility in resource allocation, the Restriction Policy must allow a larger number of pieces in the system. To this aim one can bound Card($\mathcal{I}_q$) to make a TLD and higher level deadlocks impossible "a priori". At the same time the policy must inhibit any event just leading to a FLD or to a SLD.

To go into details, let $\{\Gamma_1,\ldots,\Gamma_G\}$ be the set of rosaces of $D_{\mathcal{W}}$ corresponding to the second level cycles in Γ^2. We define the Third Level Digraph $D^3{}_{\mathcal{W}}=(\mathcal{N}^3,\mathcal{E}^3{}_{\mathcal{W}})$, by associating a vertex $n^3{}_n$ with each $\gamma^2{}_n\in\Gamma^2$, so that $\mathcal{N}^3=\{n^3{}_1,\ldots,n^3{}_G\}$. The edge $e^3{}_{vs}=(n^3{}_v,n^3{}_s)$ belongs to $\mathcal{E}^3{}_{\mathcal{W}}$, iff there exists a cycle γ_g of $D_{\mathcal{W}}$ such that $\Gamma_v\cap\Gamma_s=\{\gamma_g\}$, $\mathcal{N}_{\Gamma_v}\cap\mathcal{N}_{\Gamma_s}=\mathcal{N}_g$, $\mathcal{E}_{\Gamma_v}\cap\mathcal{E}_{\Gamma_s}=\mathcal{E}_g$ and the centres r_v of Γ_v and r_s of Γ_s are distinct.

Now let $\gamma^3{}_n=(\mathcal{N}^3{}_n,\mathcal{E}^3{}_n)$ be a cycle of $D^3{}_{\mathcal{W}}$ (Third Level Cycle) and let $\mathcal{H}_n$ be the set of rosaces in $D_{\mathcal{W}}$ corresponding to the vertices in $\mathcal{N}^3{}_n$. We define the Cycle Capacity, $C(\gamma^3{}_n)$, of the third level cycle $\gamma^3{}_n$ as the number of resources associated with the rosaces in $\mathcal{H}_n$: $C(\gamma^3{}_n)=C(\mathcal{H}_n)$. Moreover we denote by $O_q(\gamma^3{}_n)=O_q(\mathcal{H}_n)$ the Overlap Degree of $\gamma^3{}_n$ in the state q and by Γ^3 the third level cycle set.

The above ideas support the following proposition [7].

Proposition 2: Necessary condition for q to be a TLD state is that the Third Level Digraph $D^3{}_{\mathcal{W}}$ has a cycle $\gamma^3{}_n$, whose overlap degree in the state q is $O_q(\gamma^3{}_n)=[C(\gamma^3{}_n)-2]$.

Proposition 2 motivates the next deadlock-free Restriction Policy.

Restriction Policy B (RPB)

$f_1(q,w)=1$ if: $D_{Tr}(q')$ does not contain any cycle,
$O_{q'}(\gamma^2{}_n)<(C(\gamma^2{}_n)-1)$ for each $\gamma^2{}_n\in\Gamma^2$,
$\text{Card}(\mathcal{I}_{q'})<(C^3{}_0-2)$;

$f_1(q,w)=0$ otherwise;

$f_2(q,e_{im})=1$ if: $D_{Tr}(q')$ does not contain any cycle,
$O_{q'}(\gamma^2{}_n)<(C(\gamma^2{}_n)-1)$ for each $\gamma^2{}_n\in\Gamma^2$;

$f_2(q,e_{im})=0$ otherwise

where

$$C^3{}_0=\min_{\Gamma^3} C(\gamma^3{}_n)$$

and $C^3{}_0=\infty$ if $D^3{}_{\mathcal{W}}$ is acyclic.

Functions f_1 and f_2 in Restriction Policy B guarantee that the system does not reach any FLD and (by Proposition 1) any SLD state. Furthermore, f_1 bounds the number of jobs in progress so that the necessary conditions of Proposition 2 are not satisfied. Namely, we get $O_{q'}(\gamma^3{}_n)\leq\text{Card}(\mathcal{I}_{q'})<[C^3{}_0-2]\leq[C(\gamma^3{}_n)-2]$ for any $\gamma^3{}_n\in\Gamma^3$. Therefore neither a TLD nor a higher level deadlock can occur.

4 Computational Complexity

To discuss the computational complexity of the proposed Restriction Policies, we separate the on-line costs from the off-line costs. The first ones concern the real-time computations while the second ones characterize the algorithms that are employed only once, before the proper real-time control.

Restriction Policy A requires the following off-line steps: the determination of cycles from $D_{\mathcal{W}}$ and from $D^2_{\mathcal{W}}$ with their capacities, the characterization of Γ^2 and, finally, the computation of the minimum capacity C^2_0. Determining cycles of $D_{\mathcal{W}}$ and of $D^2_{\mathcal{W}}$ is the heaviest operation of this control law. To this aim, cycles can be generated in $O\{[Card(\mathcal{N})+Card(\mathcal{E})](c+1)\}$ time [10], where c represents the number of cycles of $D_{\mathcal{W}}$ or of $D^2_{\mathcal{W}}$. Obviously, the complexity of the computation is prohibitive if the number of cycles from $D_{\mathcal{W}}$ is very high (f.e. if $D_{\mathcal{W}}$ is complete). However in many applications the adjacency matrices of $D_{\mathcal{W}}$ and of $D^2_{\mathcal{W}}$ are sparse enough, so that the effort for determining cycles is reasonable.

The on-line computations of RPA consist in updating $Card(\mathcal{J}_q)$, in transforming $D_{Tr}(q)$ into $D_{Tr}(q')$ and in checking if such a new digraph contains a cycle. This last operation can be easily performed by a depth-first search algorithm whose complexity is $O[Card(\mathcal{N})]$, because the outdegree of the vertices of the Transition Digraph equals zero or one.

Restriction Policy B requires the same off-line computation steps as RPA and, in addition, it needs the determination of the digraph $D^3_{\mathcal{W}}$, of the set Γ^3 and of the minimum capacity C^3_0. For what concerns the on-line computations, RPB is more expensive than RPA. Namely, it also demands to store and update the Overlap Degree for any second level cycle from Γ^2, at each state q'.

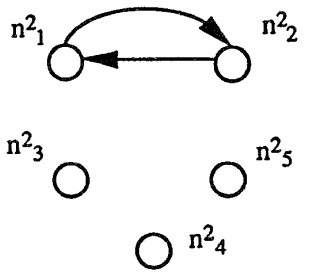
To sum up, the complexity of the off-line algorithms depends on the number of cycles from $D_{\mathcal{W}}$, $D^2_{\mathcal{W}}$ and $D^3_{\mathcal{W}}$. On the other hand, the on-line computational costs are small: thus, in many practical cases, both RPA and RPB are suitable for real-time control applications. Finally, RPB is more complex than RPA even if it is more flexible in allocating the resources at disposal.

5 Examples

This section illustrates two examples implementing the Restriction Policies defined above. In each case we also compare the Restriction Policies with the deadlock avoidance techniques proposed by Banaszack and Krogh (RPBK) [1] which is simpler and very easy to implement.

Figure 1.a and 2.a describe the Working Procedure Digraphs of Example 1 and of Example 2, respectively. Such Working Procedures have been explicitly defined in Section 2. For each Working Procedure there is a job type to produce and both examples use gamma distributions for the processing times. For what concerns loading and timing policies, all the jobs in the production mix are ordered randomly in an input queue. On each 2-type event occurrence, a job enters the system if the first resource in its Working Procedure is idle and, of course, if f_1 enables such an input. Finally, priority discipline is First In First Out.

Example 1: The production mix is composed of 50 items for each job type. The processing times are distinct in two cases. In case 1, they have mean 100 and variance 20 for each resource; in case 2 mean and variance for both r_2 and r_3 are changed to 150 and 30, respectively. While in case 2 the processing times balance the workloads of the four machines, in case 1 the system is unbalanced. Figure 3 shows the Second Level Digraph $D^2_{\mathcal{W}}$, possessing one cycle only: $\gamma^2_1=(\{n^2_1,n^2_2\}, \{e^2_{12},e^2_{21},\})$, with $C(\gamma^2_1)=C^2_0=4$. The Third Level Digraph is obviously acyclic. Concerning the established scheduling policies, the simulation assumes the makespan as performance index. The results of Figure 4 confirm that RPB is the best policy in any case.



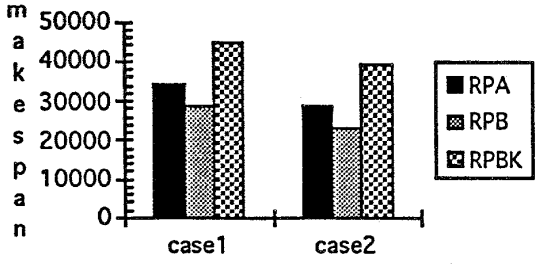
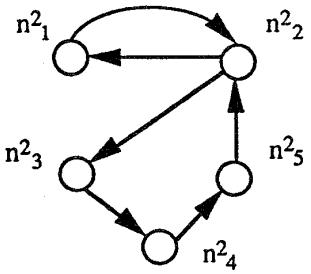


Figure 3: Digraph D^2_{Ψ} for Example 1. Figure 4: Simulation results for Example 1.

Example 2: In this example, the production mix is composed of 30 items for each job type. As in Example 1, we consider two cases: case 1 with unbalanced workload (processing times with mean 100 and variance 20) and case 2 with balanced workload (r_3 with mean 50 and variance 10 and the other resources with mean 100 and variance 20). Figure 5 shows that the digraph D^2_{Ψ} has five vertices and two cycles: $\gamma^2_1=(\{n^2_1,n^2_2\}, \{e^2_{12},e^2_{21},\})$ and $\gamma^2_2=(\{n^2_2,n^2_3,n^2_4,n^2_5\}, \{e^2_{23},e^2_{34},e^2_{45},e^2_{52}\})$, with $C(\gamma^2_1)=4$ and $C(\gamma^2_2)=5$. Moreover, we get $\Gamma^2=\{\gamma^2_1, \gamma^2_2\}$.The digraph D^3_{Ψ} has two vertices and one cycle $\gamma^3_1=(\{n^3_1,n^3_2\},\{e^3_{12},e^3_{21}\})$, with: $C(\gamma^3_1)=7$. Finally, Figure 6 shows the simulation results that lead to the same conclusions as before: the makespans obtained applying RPA and RPBK are very close to each other, while RPB gives better results.



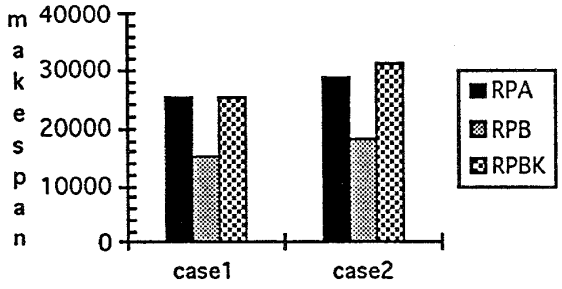


Figure 5: Digraph D^2_{Ψ} for Example 2. Figure 6: Simulation results for Example 2.

6 Conclusions

Techniques to prevent and avoid deadlock must guarantee a wide flexibility in the use of resources in FMS. Namely, reducing the allowed resource options can result in poor system performance. On the other hand, since in FMS management and control various decision making tasks are more and more assigned to on-line computers, it is acceptable to use more complex software to avoid deadlock if this leads to valuable improvements in the performance indices. In this framework, we have proposed a deadlock avoidance policy (RPB) which appears more flexible, though a little more complex, than other preceding approaches (RPA and RPBK). The results obtained by simulation confirm that RPB is less restrictive than RPA and RPBK because it allows better performance indices.

Acknowledgement
This work was supported by 40% MURST funds.

References

[1] Z.A. Banaszak and B.H. Krogh, "Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 6, pp. 724-734, Dec. 1990.

[2] C. Berge, *Graphs.* North-Holland Mathematical Library, Amsterdam, 1991.

[3] M.P. Fanti, B. Maione, G. Piscitelli, and B. Turchiano, "System approach to a generic software specification for Flexible Manufacturing System job flow management," *Int. J. of Systems Science*, vol. 23, no. 11, pp. 1889-1902, 1992.

[4] M.P. Fanti, B. Maione, G. Piscitelli, and B. Turchiano, "System Approach to Design Generic Software for Real-Time Control of Flexible Manufacturing System," *IEEE Trans. on Systems, Man, and Cybernetics*, to appear.

[5] M.P. Fanti, B. Maione, S. Mascolo, B. Turchiano "Low-cost deadlock avoidance policies for Flexible Production Systems," *Proc. of the IASTED Int. Conf. Applied Modelling and Simulation*, Lugano, Switzerland, June 20-22, 1994, pp. 219-223.

[6] M.P. Fanti, B. Maione, S. Mascolo, B. Turchiano "Event-Based Feedback Control for Deadlock Avoidance in Flexible Production Systems," DEE Report No. 12/93/S, Politecnico di Bari, Dipartimento di Elettrotecnica ed Elettronica, 1993, pp. 1-32, (submitted for publication).

[7] M.P. Fanti, B. Maione, S. Mascolo, B. Turchiano "A graph-theoretic framework for deadlock detection and avoidance in FMS," DEE Report No. 6/94/S, Politecnico di Bari, Dipartimento di Elettrotecnica ed Elettronica, 1994, pp. 1-12.

[8] F.S. Hsieh and S.C. Chang, "Dispatching-Driven Deadlock Avoidance Controller Synthesis for Flexible Manufacturing Systems," *IEEE Trans. on Robotics and Automation,* vol.10, no. 2, pp. 196-209, 1994.

[9] T.K. Kumaran, W. Chang, H. Cho, and R.A. Wysk, "A structured approach to deadlock detection, avoidance and resolution in flexible manufacturing systems", *Int. J. Prod. Res.*, vol. 32, no. 10, pp. 2361-2379, 1994.

[10] E.M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice. Prentice-Hall*, Inc. Englewood Cliffs, New Jersey, 1977.

[11] R.A. Wysk, N.S. Yang, and S. Joshi, "Detection of Deadlocks in Flexible Manufacturing Cells," *IEEE Trans. on Robotics and Automation*, vol. 7, no.6, pp. 853-859, Dec. 1991.

[12] N. Viswanadham , Y. Narahari, and T.L. Johnson, "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 6, pp. 713-723, Dec. 1990.

[13] B.P. Zeigler, *Multifacetted Modelling and Discrete Event Simulation.* Academic Press Inc., London, 1984.