Digraph-Theoretic Approach for Deadlock Detection and Recovery in Flexible Production Systems

Maria Pia Fanti, Guido Maione, Biagio Turchiano
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari
Via Re David, 200
70125 Bari
ITALY

Abstract: In flexible manufacturing systems a deadlock arises when jobs in a set are indefinitely prevented from accessing resources because these are taken by other jobs in the same set. This condition is highly unfavourable because it stops the normal flow of parts and can propagate to the entire system. To face this problem, recent literature proposes prevention, avoidance and detection/recovery techniques. This paper introduces a detection/recovery approach based on a simple digraph that characterizes deadlock by describing the current interactions between pieces and resources. The method requires a low computation burden in the detection phase and makes use of a dedicated buffer to activate the recovery phase. Finally, a case study shows the simplicity and effectiveness of the proposed approach.

Keywords: Deadlock, Manufacturing Automation, Flexible Manufacturing Systems

Maria Pia Fanti was born at Siena, Italy, in 1957. She received the degree in Electronic Engineering from the University of Pisa, Italy in 1983. She is currently Assistant Researcher at the Department of Electrical and Electronic Engineering, the Polytechnic of Bari, Italy. Her research focuses on structural properties of linear systems and on FMS control and modelling.

Guido Maione was born in Naples, on August 10, 1967. He received the degree in Electronic Engineering with honours from the Polytechnic of Bari in 1992. He is currently a Ph.D student at the Department of Electrical and Electronic Engineering, the Polytechnic of Bari. His research interests are in the areas of FMS control and modelling, object-oriented control of FMS.

Biagio Turchiano was born at Bitetto, Bari, Italy, on July 25, 1953. He received the degree in Electrical Engineering with honours from the University of Bari in 1979. He joined the Department of Electrical and Electronic Engineering, Polytechnic of Bari in 1984 as Assistant Researcher. He is currently Associate Professor of Automatic Control. His research and teaching interests are in the areas of production automation, systems and control theory, modelling and control of discrete event systems.

1. Introduction

Flexible Manufacturing Systems (FMSs) consist of a set of resources (workcentres, measure and inspection centres, buffers, transport devices, etc.) which the pieces (jobs, items) request service from. In these systems competition of jobs for a limited number of resources can lead to deadlock situations (circular waits). These occur when jobs from a set that holds resources are blocked indefinitely from access to the resources held by other jobs within the same set. Deadlock is dangerous because it stops production and can propagate to the entire system.

The problem has been extensively studied in computer science for multitasking environments [2]. Only recently have some authors focused on the deadlock in FMSs. Namely there are many peculiarities in deadlock analysis of manufacturing systems, e.g. processing times of pieces are considerably longer than computation times. Moreover the sequence in which each job requests the resources is exactly known in FMSs, so that one can profitably use this information in deadlock-solving algorithms. Hence, due to these peculiarities, it is necessary to develop special techniques for dealing with deadlocks in FMSs.

As in the computer science context, the methodologies to face deadlock in FMSs can be classified in three categories: prevention, avoidance and detection/recovery. The first two classes rule the flow of jobs so as to prevent the system from reaching deadlock conditions [1, 4-7, 9, 13, 15]. In particular, the former utilizes static policies, the latter uses information on the current state of the FMS to limit the freedom in resource allocation. The detection/recovery approaches do not limit in any way the freedom in allocation of resources to jobs [10, 14, 15]. Namely, they only identify current deadlock situations (detection phase) and, if any, start special policies of resource management to break the condition of circular wait and restore normal operating conditions (recovery phase).

Existing techniques of prevention and avoidance impose constraints on resource allocation that can determine reduced utilization of the resources and poor performance of the production system. Generally speaking, even if they involve higher on-line computational burden, the constraints for deadlock avoidance techniques are less restrictive than prevention methods. On the contrary detection/recovery approaches do not constrain the resources during the normal system operation. However they involve additional costs due to both the hardware necessary to activate the recovery policy and the system performance reduction throughout the recovery phase. Consequently the choice of the more favourable method to face deadlock depends on the peculiarities and on the lay-out of the production system under consideration.

This paper analyzes the deadlock in FMSs by using a simple model of the interactions between jobs and non-unitary capacity resources, i.e. resources with multiple items of the same type. The underlying idea is the definition of a digraph named Transition Digraph that describes the process of resource acquiring and releasing, modelled as a Discrete Event Dynamical System (DEDS). In particular, the Transition Digraph captures the main information of the DEDS state and changes with time as state evolves. The paper shows that there is an equivalence between deadlock and a figure in the transition digraph, named Maximal-weight Zero-outdegree Strong Component (MZSC). This formal characterization of the deadlock allows us to show the effectiveness of a detection procedure identifying the MZSCs by a depth-first search on the Transition Digraph. On the basis of this method and of a recovery mechanism using a reserved central buffer of unit-capacity, we develop an effective detection /recovery policy.

The paper is organized in six Sections. Section 2 describes the peculiarities of the DEDS modelling the production system and defines the Transition Digraph. Moreover it indicates the laws ruling the Transition Digraph modifications according to the state changes and triggered by events involving acquisition or release of resources. Section 3 proves the equivalence between deadlock and MZSC in the Transition Digraph, while Section 4 illustrates the detection/recovery policy. Finally, Section 5 shows the application of such a policy to a case study and Section 6 draws the conclusions.

2. The Model

This section describes the peculiarities of the DEDS modelling the production system and defines the Transition Digraph.

2.1 The Discrete Event Dynamical System

Let us first look at some background notations that characterize the model of a production system S. The symbol $R=\{r_i, i=1, 2,..., R\}$ indicates the set of resources, where the first R-1 elements represent multiple slot buffers, machines with identical servers, AGV systems provided with one or several trucks, etc. while r_R is an additional fictitious resource the jobs acquire as they leave the system. An integer $C(r_i)$ indicates the capacity of the resource r_i, i.e. the maximum number of jobs that can contemporaneously hold r_i. Thus $C(r_i)$ is finite for each resource, but for r_R that can receive jobs without limits. Now if J is the set of jobs to produce, each element from J requires a sequence of resources, named Working Procedure. Thus the set $W=\{\mathbf{w}\}$ collects the Working Procedures necessary for processing all the jobs from J. Obviously the terminal resource of each $\mathbf{w}\in W$ is a fictitious one. With these preliminary notions as background, we refer to a DEDS model to describe the process thereby jobs acquire or release resources [3]. The state of the DEDS model $\mathbf{q}$ contains information on the operating conditions of S, such as the set $J_\mathbf{q}$ of jobs in process, the resources currently held by each job from $J_\mathbf{q}$, the Working Procedures associated with such jobs, and, finally, the Residual Working Procedures, i.e. the resources necessary for each $j\in J_\mathbf{q}$ to complete its processing. The set of all the DEDS states is denoted by Q.

Now, with reference to the current state $\mathbf{q}$, we introduce some additional notations. In particular HR(j) denotes the resource currently held by $j\in J_\mathbf{q}$, SR(j) identifies the second resource of the Residual Working Procedure of such a job and WP(j) indicates its complete Working Procedure. Note that SR(j) is defined in all the cases. Namely, any Residual Working Procedure contains r_R and, by assumption, a job leaving the system accedes to r_R, but, at the same time, it is removed from the set $J_\mathbf{q}$. Finally we say that a resource r_i is *empty* (*idle* or *busy*) in the state $\mathbf{q}$, if $n_i=0$ ($0<n_i<C(r_i)$ or $n_i=C(r_i)$ respectively) where n_i denotes the number of jobs holding such a resource.

Obviously, the DEDS must encompass events involving resource acquiring or releasing. We consider two event types:

a) a new job enters the system (1-type event). This event is identified by a pair (j,**w**), where $j\in J$ is the job entering S and $\mathbf{w}\in W$ is the Working Procedure the job has to follow;

b) a job progresses from a resource to another one, or it leaves the system (2-type event). This event is specified by a job $j\in J_\mathbf{q}$ progressing from HR(j) to SR(j), where $\mathbf{q}$ is the current state of S.

2.2 The Transition Digraph

Deadlock detection requires the description of the current interactions between jobs and resources, in each state $\mathbf{q}$ of the DEDS. To this aim, we introduce a digraph, named Transition Digraph and denoted by $D_{Tr}(\mathbf{q})=[N, E_{Tr}(\mathbf{q})]$. The vertex set represents the system resource set. So, for the sake of simplicity, the same symbol indicates the elements of the node set N and the system resources, i.e. $N=R$. Moreover, the edge set changes as $\mathbf{q}$ is updated: an edge e_{im} is in $E_{Tr}(\mathbf{q})$ if and only if (iff for brevity) a job $j \in J_\mathbf{q}$ holds r_i in the state $\mathbf{q}$ and requires r_m as next resource. Therefore the Transition Digraph indicates both the resources currently held by jobs from $J_\mathbf{q}$ and the resources required by the same jobs in the next step of their Working Procedures.

We note that a single edge $e_{im} \in E_{Tr}(\mathbf{q})$ may represent more jobs detaining r_i and requesting r_m. To take into account this situation, we associate the following weight with each edge $e_{im} \in E_{Tr}(\mathbf{q})$:

$$a_\mathbf{q}(e_{im}) = \text{Card}\{j \in J_\mathbf{q}: \text{HR}(j)=r_i \text{ and } \text{SR}(j)=r_m\} \quad (1)$$

where Card(.) stands for "cardinality of …". In this way, $a_\mathbf{q}(e_{im})$ yields the number of jobs which hold r_i and request r_m as next resource.

Now, according to Harary *et al* [8], we define Outdegree Value of a node r_i as the sum of weights of edges outgoing from r_i in $D_{Tr}(\mathbf{q})$, i.e.

$$OV_\mathbf{q}(r_i) = \sum_{m=1}^{R} a_\mathbf{q}(e_{im}) \quad (2)$$

where we consider $a_\mathbf{q}(e_{im})=0$ if $e_{im} \notin E_{Tr}(\mathbf{q})$. Hence, the Outdegree Value of a vertex indicates the number of jobs currently using the corresponding resource in the state $\mathbf{q}$. In particular, if $OV_\mathbf{q}(r_i)=0$ then r_i is *empty;* if $0<OV_\mathbf{q}(r_i)<C(r_i)$ then r_i is *idle*; finally, if $OV_\mathbf{q}(r_i)=C(r_i)$ then r_i is *busy*. Obviously, a job $j \in J_\mathbf{q}$ requiring r_m as next resource, is *blocked* iff $OV_\mathbf{q}(r_m)=C(r_m)$. On the contrary, if $OV_\mathbf{q}(r_m)<C(r_m)$ then job j is *unblocked*. We indicate by $J_{\mathbf{q},u} \subset J_\mathbf{q}$ the set of jobs unblocked in the state $\mathbf{q}$.

Before continuing, let us show how to update the Transition Digraph at each event occurrence. Suppose S be in the state $\mathbf{q}$ and consider a Working Procedure $\mathbf{w} \in W$. In such a condition for a job $j \in J$ to enter S and to receive service according to $\mathbf{w}$, the first resource in $\mathbf{w}$ must necessarily be *idle* or *empty*. On the occurrence of this 1-type event, S makes transition from $\mathbf{q}$ to a new state $\mathbf{q}'$. The new Transition Digraph $D_{Tr}(\mathbf{q}')$ is obtained as follows: if r_m and r_p are respectively the first and the second resource in $\mathbf{w}$, then $E_{Tr}(\mathbf{q}) \cup \{e_{mp}\}$ yields the edge set $E_{Tr}(\mathbf{q}')$. Clearly it holds: $OV_{\mathbf{q}'}(r_m)=OV_\mathbf{q}(r_m)+1$.

Analogously, let $j \in J_{\mathbf{q},u}$, $r_i=\text{HR}(j)$ and $r_m=\text{SR}(j)$. By definition of the set $J_{\mathbf{q},u}$, r_m is *idle* or *empty* in the state $\mathbf{q}$, so that the transition leading j from r_i to r_m can occur. This 2-type event updates the state from $\mathbf{q}$ to $\mathbf{q}'$. In particular, the edge set $E_{Tr}(\mathbf{q}')$ of $D_{Tr}(\mathbf{q}')$ and the corresponding edge weights result from the following operations on $E_{Tr}(\mathbf{q})$:

i) put $a_{\mathbf{q}'}(e_{im})=a_\mathbf{q}(e_{im})-1$ and, if $a_\mathbf{q}(e_{im})=1$, remove e_{im} from $E_{Tr}(\mathbf{q})$;

ii) provided that $\text{SR}(j) \neq r_R$, put $a_{\mathbf{q}'}(e_{mp})=a_\mathbf{q}(e_{mp})+1$, where r_p is the third resource in the Residual Working Procedure of job j, and, if $a_\mathbf{q}(e_{mp})=0$, put $E_{Tr}(\mathbf{q}') = E_{Tr}(\mathbf{q}) \cup \{e_{mp}\}$.

In this way, r_i becomes *empty* if $OV_\mathbf{q}(r_i)=1$ or *idle* if $OV_\mathbf{q}(r_i)>1$, while r_m becomes *busy* if $OV_\mathbf{q}(r_m) = C(r_m)-1$ or *idle* if $OV_\mathbf{q}(r_m) < C(r_m)-1$.

Of course all the remaining nodes keep unchanged their Outdegree value and the *busy/idle/empty* condition they had in $D_{Tr}(\mathbf{q})$.

3. Detecting Deadlock By the Transition Digraph

This Section proves a result that allows deadlock detection on the basis of some particular strong components of the Transition Digraph. For the definition of strong component, walk and other standard figures of digraphs, we refer to [8].

Now, let us begin with some preliminary ideas. As mentioned before, $\mathbf{q}$ is a deadlock state if each member of a job set waits indefinitely for other jobs in the same set to release resources. The following definition expresses this condition more formally.

Definition 1: $\mathbf{q} \in Q$ is a deadlock state for S if there exist a non-empty job subset $J_D \subset J_\mathbf{q}$ and a non-empty resource subset $R_D \subset R$, satisfying the following properties:

D1a) J_D is the maximal subset of $J_\mathbf{q}$ such that $\text{HR}(J_D)=R_D$;

D1b) $\text{SR}(J_D) \subset R_D$;

D1c) any resource in $\text{SR}(J_D)$ is *busy*.

Previous definition has a quite transparent meaning. Namely, by D1c) each job from the set J_D is *blocked* and, by D1a) and D1b), it requires a resource held by other jobs in J_D. Deadlock conditions are related to some particular strong components of the digraph $D_{Tr}(\mathbf{q})$ characterized by the following definition.

Definition 2: Let $\sigma=(N_\sigma, E_\sigma)$ be a strong component of $D_{Tr}(\mathbf{q})$. We call σ a *"Maximal-weight Zero-outdegree Strong Component"* of $D_{Tr}(\mathbf{q})$ (MZSC for brevity) if it enjoys the following properties:

D2a) *Maximal-weight:* all the resources from N_σ are *busy*, i.e. $OV_\mathbf{q}(N_\sigma)=C(N_\sigma)$;

D2b) *Zero-outdegree:* all the edges of $D_{Tr}(\mathbf{q})$ outgoing from vertices of N_σ belong to E_σ, i.e. the elements from N_σ are the only vertices of $D_{Tr}(\mathbf{q})$ reachable from vertices in N_σ.

In the above statement, symbols $OV_\mathbf{q}(N_\sigma)$ and $C(N_\sigma)$ indicate the sum of $OV_\mathbf{q}(r_i)$ and $C(r_i)$ over all the vertices from the set N_σ, and constitute the Overlap Degree and the Capacity of σ, respectively. The following result gives a necessary and sufficient condition for the occurrence of a deadlock state in a system with multiple capacity resources.

Proposition 1: $\mathbf{q}$ is a deadlock state for S iff there exists at least one MZSC in $D_{Tr}(\mathbf{q})$.

Proof

Comparing conditions D1a), D1b) and D1c) for the deadlock occurrence (see Section 2) with the properties of an MZSC gives the key to prove this proposition.

If part

Let $\sigma=(N_\sigma, E_\sigma)$ be an MZSC of $D_{Tr}(\mathbf{q})$ and let J_D indicate the maximal subset of $J_\mathbf{q}$ such that $HR(J_D)=N_\sigma$. The set J_D certainly exists since, by D2a), all the resources from N_σ are *busy*. On the other hand, by D2b) all the edges outgoing from vertices of N_σ terminate in nodes still belonging to N_σ. This implies: $SR(J_D)\subset N_\sigma$. Hence putting $R_D=N_\sigma$ verifies conditions D1a), D1b) and D1c).

Only if part

The proof is in three steps.

Step 1 By condition D1a) at least one edge of $D_{Tr}(\mathbf{q})$ starts from each node of R_D. Moreover, by D1b) all the edges originating from vertices of R_D still terminate in R_D. Consequently there exists a walk [8] of infinite length starting from each vertex in R_D and touching nodes from R_D only. Since the cardinality of R_D is finite, $D_{Tr}(\mathbf{q})$ must contain at least one non-trivial strong component with all its vertices in R_D. Moreover each node from R_D is the starting point of a walk reaching one of these strong components. Obviously such components are disjoint and two vertices from two different components can never be mutually reachable. Hence, at least one of them, say $\sigma=(N_\sigma, E_\sigma)$, cannot reach any one of the remaining strong components having vertices in R_D.

Step 2 We now prove that all the edges of $D_{Tr}(\mathbf{q})$ outgoing from N_σ belong to E_σ. The proof is by contradiction. Suppose there exists a vertex $r_m \notin N_\sigma$, reachable from some node in N_σ. By step 1, $r_m \in R_D$ and there exists a strong component of $D_{Tr}(\mathbf{q})$ having vertices in R_D and which is reachable from r_m. Let σ^* be such a component. If $\sigma^*=\sigma$ then r_m and vertices from N_σ are mutually reachable. But this conclusion leads to the contradiction: $r_m \in N_\sigma$. On the other hand, if $\sigma^* \neq \sigma$, the fact that vertices from σ^* are reachable from nodes of σ contradicts the definition of σ given at Step 1. To sum up, this step proves D2b) for σ.

Step 3 By condition D1a), if for any job $j \in J_\mathbf{q}$ it holds $HR(j) \in N_\sigma$, then $j \in J_D$. Moreover, since each vertex from N_σ is adjacent from another vertex of N_σ, we get: $N_\sigma \subset SR(J_D)$. Hence condition D1c) implies that each vertex from N_σ is *busy* . This proves D2a).

Example 1

To illustrate Proposition 1 and clarify the notations, we consider a system with R=4, producing a job mix J according to the following Working Procedures:

$$w_1 = (r_3, r_1, r_3, r_2, r_4)$$

$$w_2 = (r_3, r_2, r_3, r_4)$$

Let the capacities of the resources, but the fictitious one, be $C(r_i)=2$ for $i=1,\ldots,3$. Moreover let the system be in a state $\mathbf{q}$, with: $J_\mathbf{q}=\{j_i: i=1, 2,\ldots, 6\}$; $HR(j_1) = HR(j_2) = r_3$, $SR(j_1) = r_2$ and $SR(j_2) = r_1$; $HR(j_3) = HR(j_4) = r_1$ and $SR(j_3) = SR(j_4) = r_3$; $HR(j_5) = HR(j_6) = r_2$ and $SR(j_5) = SR(j_6) = r_3$.

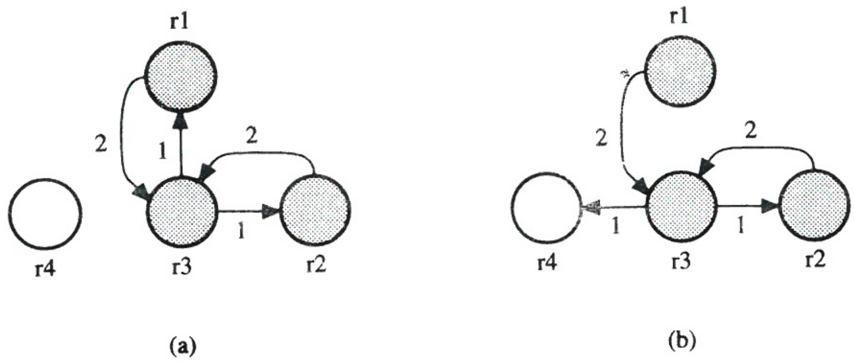

Figure 1. (a): $D_{Tr}(q)$ exhibits a deadlock condition; (b): $D_{Tr}(q')$ contains no MZSC

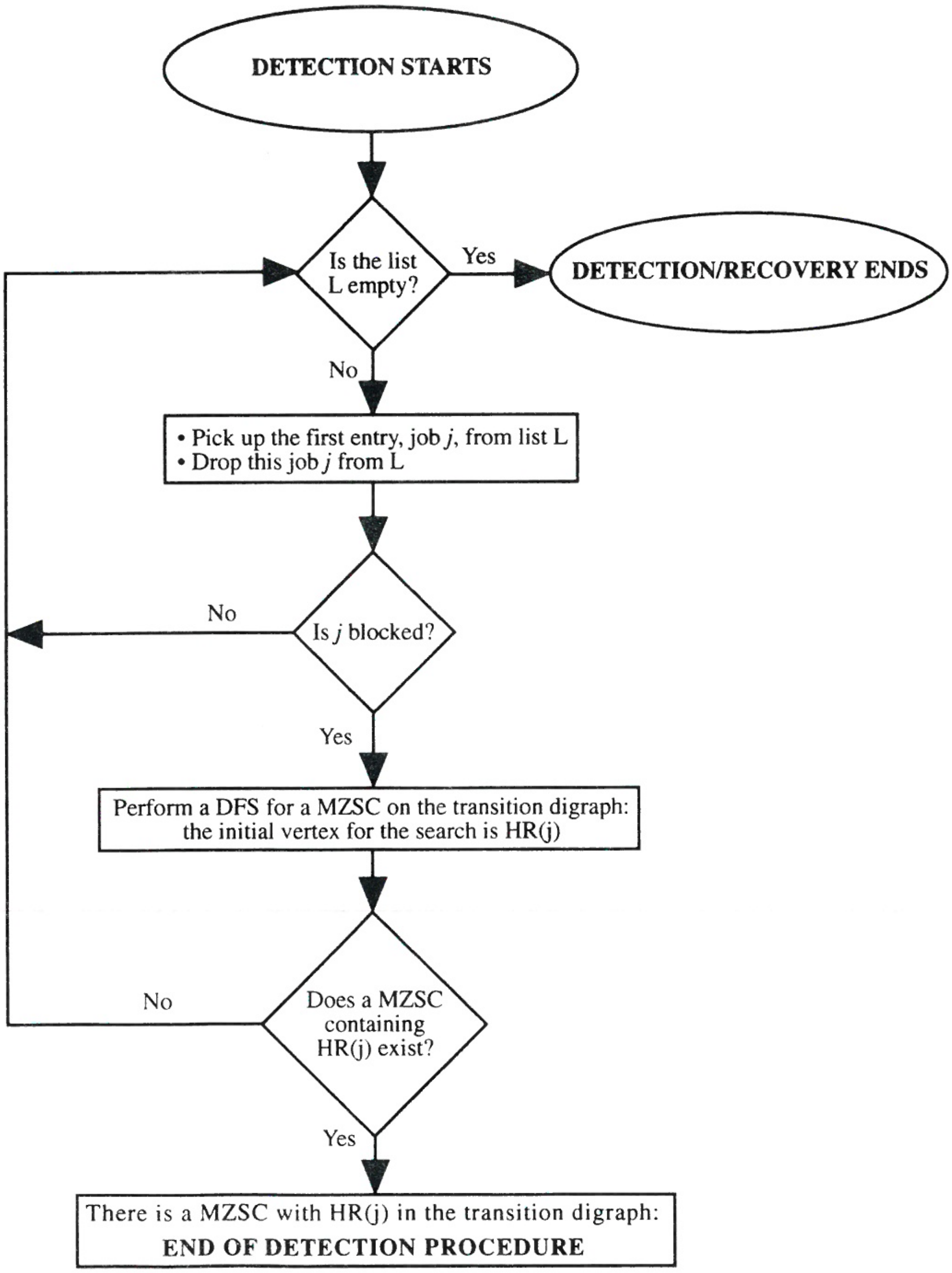

Figure 2. Detection Procedure Flow-chart

Figure 1a shows the Transition Digraph $D_{Tr}(\mathbf{q})$ where edges are labelled by the corresponding weights and dark nodes indicate *busy* resources. The strong component σ=({r_3, r_1, r_2}, {e_{31}, e_{13}, e_{32}, e_{23}}) is a MZSC in the deadlock state **q**. It is easy to verify that all the jobs in $J_\mathbf{q}$ are permanently blocked, i.e. **q** is a deadlock state. On the contrary, for the state **q'** equal to **q** but for $SR(j_1)=r_4$, no MZSC exists in $D_{Tr}(\mathbf{q'})$ (see Figure 1b). State **q'**, indeed, is not a deadlock state because no job from $J_\mathbf{q}$ is indefinitely blocked on the resource it currently holds.

Proposition 1 suggests an easy, real-time deadlock detection procedure. To apply it, the controller has to keep memory of the Transition Digraph and to update it according to the rules described at the end of the previous Section. In particular, after each 1-type or 2-type event occurrence, the controller must check if the Transition Digraph contains any MZSC. If this is the case, the controller determines the sets of parts and of resources involved in the deadlock. Then it triggers the recovery procedure that moves a deadlocked job into a special storage buffer B of unit-capacity. This action breaks the circular wait condition and, if combined with a proper policy for resource allocation, resolves the deadlock. The following Section describes these concepts in detail.

4. Detection/Recovery Algorithm

As mentioned in the previous section, the algorithm is in two phases called Detection Procedure and Recovery Procedure, respectively.

4.1 Detection Procedure

When a job requires a *busy* resource, its identifier is added to a list (L) containing all the blocked jobs. Then, if the Detection/Recovery algorithm is still in execution no more actions are required at the moment. On the other hand, if the algorithm is not in execution, the Detection Procedure gets started following the steps below (see Figure 2).

1) If L is empty, the Detection/Recovery algorithm comes to an end. On the contrary, if L is not empty, then its first entry (say j) is considered and dropped from the list.
2) If j is no longer a blocked job, step 1) is executed again. On the other hand, if j is still blocked, a Depth-First Search (DFS for brevity) on $D_{Tr}(\mathbf{q})$ begins to check if a MZSC containing the vertex HR(j) exists.
3) If no MZSC containing HR(j) exists, the procedure executes step 1) again. On the contrary, if the DFS finds such a component σ, detection ends while the Recovery Procedure starts.

Obviously, the DFS is the core of the detection procedure. In particular, to determine the MZSC we use the algorithm proposed in [12] that finds strongly connected components. Such an algorithm is suitably adapted to our specific problem as described in the schema of Figure 3. Indeed it searches for a strong component of $D_{Tr}(\mathbf{q})$ that must: (a) have zero outdegree, (b) contain vertex HR(j), (c) contain only busy vertices.

It is well- known that in each digraph there is always a strong component (eventually trivial) with zero outdegree. Moreover it can be easily verified that the first strong component determined by the algorithm proposed in [12] has zero outdegree. So, choosing HR(j) as starting vertex for the DFS, the first strong component the algorithm finds (say σ) is just the zero-outdegree strong component containing HR(j), if it exists. In this case, HR(j) coincides with the root of σ [12]. So, when the DFS determines the first strong component, the algorithm checks if HR(j) is its root: if this is the case, conditions (a) and (b) are enjoyed. Moreover, condition (c) is checked for each new vertex considered by the DFS.

We remark that the proposed depth-first search algorithm is performed in $O(e)$ time, where e is the number of edges in the Transition Digraph.

Figure 4 depicts a DFS example: a job holding r_2 requests the busy resource r_3, so that r_2 is the initial vertex for the DFS (see Figure 4(a)). The steps of the procedure are indicated by the vertices and the edges examined in the search, up to the MZSC identification (see Figure 4(i)). We suppose that all the five resources are busy. All *tree* edges (continuous lines), *back* edges (dashed lines) and *forward* edges (short dashed lines) [12] are labelled by their weights. Figure 4 also shows the DFS numbers (bold numbers) and the lowlink numbers (bold numbers in parentheses) [12]. In the final step the MZSC itself is indicated.

4.2 Recovery Procedure

Once the detection finds an MZSC, the Recovery Procedure starts according to the following steps (see Figure 5).

```
S ← empty stack
i ← 0
for x ∈ V do num(x) ← 0
r ← HR(j)
[r is the initial vertex for the Depth-First Search]
STRONG(r)
procedure STRONG(v)
  if v is empty or idle then
      [There is no MZSC]
      Stop the algorithm
  else
      i ← i + 1
      num(v) ← i
      lowlink(v) ← i
      S ⇐ v
      for w ∈ Adj(v) do
          if num(v) = 0 then
              [(v, w) is a tree edge]
              STRONG(W)
              lowlink(v) ← min{lowlink(v), lowlink(w)}
          else if num(w) < num(v) then
              [(v, w) is a back edge or a cross edge]
              if w is on S then
                  [w is in the same strong component as v,
                  since w ∈ Adj(v) and since w on S implies
                  that there is a path from w to v]
                  lowlink(v) ← min{lowlink(v), num(w)}
      if lowlink(v) = num(v) then
          [There is a strong component:
          its root is v]
          if v = r then
              [r is the root of the strong component
              and this component is a MZSC]
              while x, the top vertex on S, satisfies:
                  num(x) ≥ num(v) do
                      Add x to the MZSC
                      Delete x from S
          else
              [r is not the root of the strong component,
              so there is no MZSC containing r]
              Stop the algorithm
return
```

Figure 3. DFS Algorithm

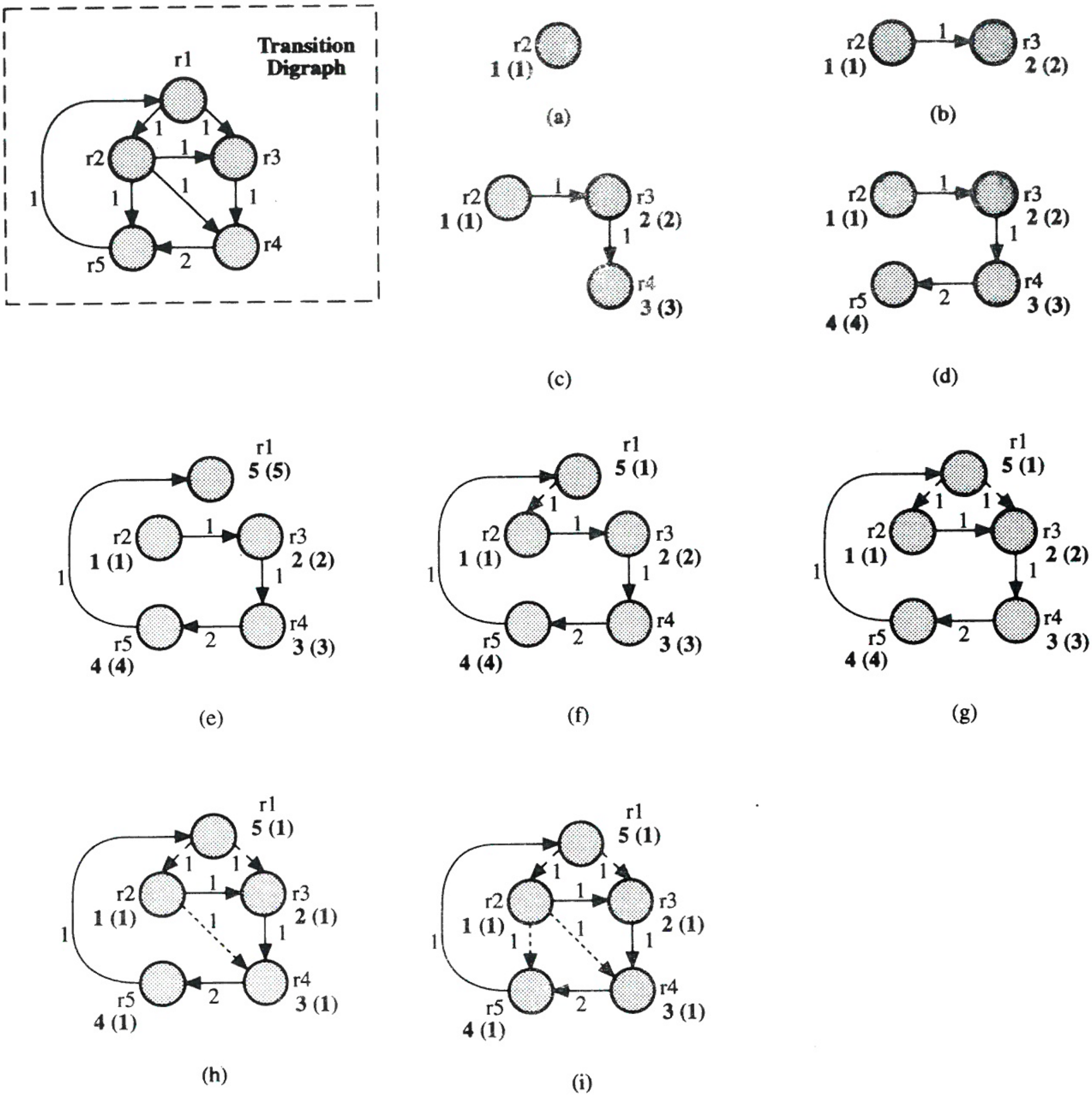

Figure 4. A DFS Example

1) Identify a *cycle* $\gamma_n=(N_n, E_n)$ in the MZSC σ, containing edge e_{im}, where r_i=HR(j) and r_m=SR(j). A DFS inside the MZSC performs this identification, by choosing e_{im} as first edge. A DFS algorithm generating all the cycles of a digraph, performs this operation in time $O[(e+v)(c+1)]$, where v is the number of nodes in the digraph and c is the number of generated cycles [12]. However, the algorithm has to determine one cycle only, by using r_i as first vertex and e_{im} as first edge in the search. So, it is executed in time $O(e+v)$.
2) Move job j to the buffer B. In this way the system state becomes **q'**, with: $OV_{q'}(r_i) = C(r_i) - 1$ and $a_{q'}(e_{im}) = a_q(e_{im}) - 1$. Obviously $D_{Tr}(q')$ does not contain σ as an MZSC so that a transition of any job to r_i becomes feasible.
3) Impose a restriction on resource allocation preventing any arrival of additional jobs to resources in N_n. This policy must inhibit both new jobs from entering the system to reach resources in N_n and all job transitions corresponding to edges e_{kp}, with $r_k \notin N_n$ and $r_p \in N_n$. Obviously, in this way jobs holding resources from N_n and requiring resources in the same set can go one step ahead in their Residual Working Procedures.

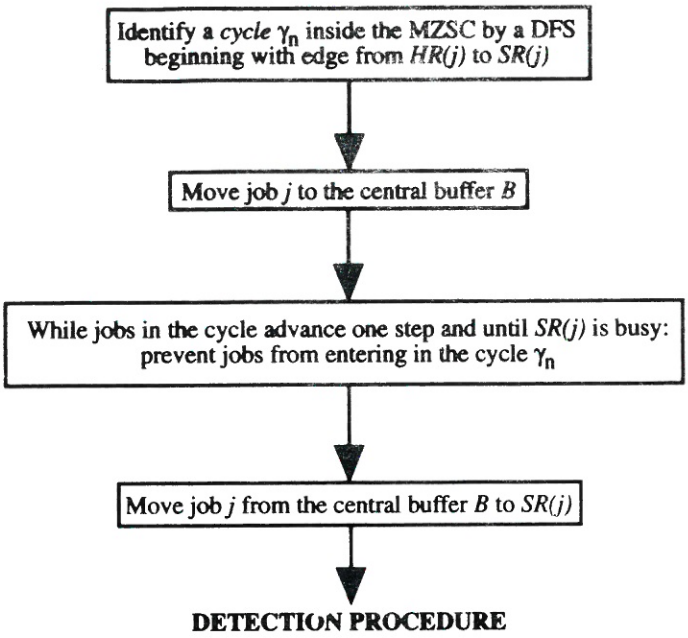

Figure 5. The Deadlock Recovery Procedure Flow-chart

4) When the system reaches a state **q*** such that $OV_{q^*}(r_m) = C(r_m) - 1$, job j is transferred from B to r_m and the restriction policy is removed. So, the Recovery ends and the Detection Procedure re-starts.

5. A Case-Study

This section applies the deadlock detection/recovery technique described above to the flexible manufacturing cell shown in Figure 6.

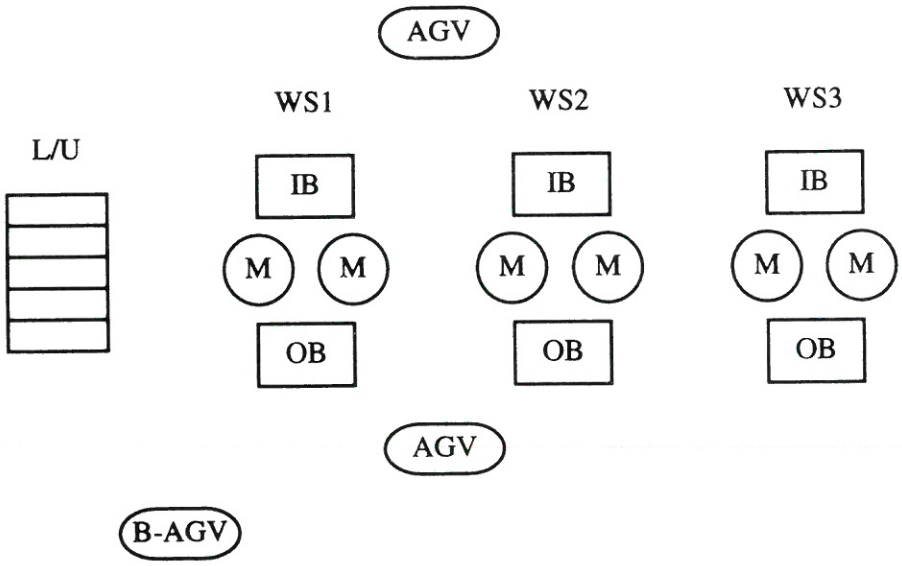

Figure 6. System in the Case-Study

Note that at the end of the recovery there may still be a MZSC in the resulting Transition Digraph. In this case a new execution of the Detection/Recovery algorithm is necessary. This allows some jobs involved in the deadlock to proceed one step more ahead in their Residual Working Procedure, so that no piece can remain indefinitely blocked.

The system consists of a load/unload station (L/U for brevity), three workstations (WSs) and an AGV system with two trucks. Each WS is composed of two identical machines (M) and has an input buffer (IB) and an output buffer (OB), both of unit-capacity. The load/unload station can take no more than five pieces. Thus Table I lists R=12 distinct resources, including the fictitious one.

Table I. Resources and Their Capacities

resources		capacities
r_1	load/unload station	$C(r_1)=5$
r_2	input buffer of WS1	$C(r_2)=1$
r_3	input buffer of WS2	$C(r_3)=1$
r_4	input buffer of WS3	$C(r_4)=1$
r_5	machines of WS1	$C(r_5)=2$
r_6	machines of WS2	$C(r_6)=2$
r_7	machines of WS3	$C(r_7)=2$
r_8	output buffer of WS1	$C(r_8)=1$
r_9	output buffer of WS2	$C(r_9)=1$
r_{10}	output buffer of WS3	$C(r_{10})=1$
r_{11}	AGV system	$C(r_{11})=2$
r_{12}	fictitious resource	∞

There are two job types to produce: the former requires a machine of WS1 and, successively, a machine of WS2; the latter receives service from a machine of WS1 and then from a machine of WS3. The AGV units transfer pieces from the L/U to the IBs and from the OBs to either the IBs or the L/U. Moreover a third AGV truck, denoted by B-AGV, plays the role of the reserved buffer B necessary to carry out the recovery procedure. Going into detail we get:

$$w_1 = (r_1, r_{11}, r_2, r_5, r_8, r_{11}, r_3, r_6, r_9, r_{11}, r_1, r_{12})$$

$$w_2 = (r_1, r_{11}, r_2, r_5, r_8, r_{11}, r_4, r_7, r_{10}, r_{11}, r_1, r_{12})$$

We implement the described detection/recovery procedure by a SIMAN discrete-event simulation model [11] assuming the system throughput as performance index. The following conditions rule the simulation. Job types enter the system according to a randomly generated sequence, with equal probability for each type. Since each simulation is performed with a constant number N of jobs in process, a new piece is loaded as soon as a completed job leaves the system. Service times for machines and AGV are generated by a gamma distribution with mean m (reported in Table II) and standard deviation s=40% of m (case A) or s=80% of m (case B). Finally, the law "First In First Out" rules the priority setting.

Table II. Mean of Service Times

resources	means
r_1	10
r_5	40
r_6	30
r_7	30
r_{11}	10

Table III shows the throughputs (jobs per time unit) resulting from each simulation of 1000 completed parts, for different values of number N. The second column of Table II reports the number of times (R) the recovery procedure is triggered in each simulation.

Table III. Simulation Results

N	Case A		Case B	
	R	Throughput	R	Throughput
3	0	.0245	0	.0244
4	0	.0319	0	.0308
5	0	.0383	0	.0368
6	7	.0434	24	.0404
7	36	.0459	63	.0436
8	**86**	**.0470**	149	.0433
9	213	.0462	**195**	**.0440**
10	317	.0455	290	.0416
11	439	.0448	400	.0412
12	551	.0443	570	.0403
13	749	.0430	770	.0400
14	1017	.0447	1037	.0386
15	1389	.0440	1371	.0397
16	1729	.0446	1678	.0398

The results show that for N=3, 4, 5 the recovery policy is not invoked because no deadlock occurs (R=0). The throughput reaches the highest values in Case A for N=8 and in Case B for N=9 (shown in bold by Table III). For N>8 in Case A and N>9 in Case B, throughput decreases because of congestion conditions due to the high work in process. The results of Table III confirm that the deadlock/recovery technique allows us to increase the number of pieces in the system and to obtain better performance indices.

6. Concluding Remarks

The detection/recovery method proposed in Section 4 appears to be simple and effective. In particular, the detection phase requires low computational burden because the method developing the idea of MZSC and using an DFS-like algorithm involves linear complexity.

The recovery phase is also very simple and it only requires a dedicated unit-capacity buffer.

These characteristics make the proposed approach easy to apply at low costs. On the other hand, as shown by the case- study, the technique allows us to increase the system utilization, thus improving its performance indices and avoiding the unfavourable effects of deadlock at the same time.

Acknowledgment

This work was supported by Italian National Council for Researches under the contract CT 96-00056.

REFERENCES

1. BANASZAK, Z.A. and KROGH B.H., **Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows**, IEEE TRANS. ROBOTICS AND AUTOMATION, Vol. 6, No. 6, December 1990, pp. 724-734.
2. COFFMAN, E.G., ELPHICK, M.J. and SHOSHANI, A., **System Deadlocks**, COMPUTING SURVEYS, Vol. 3, No. 2, June 1971, pp. 67-78.
3. FANTI, M.P., MAIONE, B., PISCITELLI, G. and TURCHIANO, B., **System Approach To Design Generic Software for Real-time Control of Flexible Manufacturing System**, IEEE TRANS. SYSTEMS, MAN AND CYBERNETICS - PART A: SYSTEMS AND HUMANS, Vol. 26, No. 2, March 1996, pp. 190-202.
4. FANTI, M.P., MAIONE, B., MASCOLO, S. and TURCHIANO, B., **Event-Based Feedback Control for Deadlock Avoidance in Flexible Production Systems**, to be published in IEEE TRANS. ROBOTICS AND AUTOMATION.
5. FANTI, M.P., MAIONE, B., MASCOLO, S. and TURCHIANO, B., **Low-cost Deadlock Avoidance Policies for Flexible Production Systems**, Proc. of the IASTED Int. Conf. Applied Modelling and Simulation, Lugano, Switzerland, June 1994, pp. 219-223, to appear in INT. J. MODELLING AND SIMULATION.
6. FANTI, M.P., MAIONE, B., MASCOLO, S. and TURCHIANO, B., **Performance of Deadlock Avoidance Algorithms in Flexible Manufacturing Systems**, JOURNAL OF MANUFACTURING SYSTEMS, Vol. 15, No. 3, 1996, pp. 164-178.
7. FANTI, M.P., MAIONE, B. and TURCHIANO, B., **Deadlock Avoidance in Flexible Production Systems with Multiple Capacity Resources**, Report N. 28/95/S, Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 1995, pp. 1-22.
8. HARARY, F., NORMAN, R.Z. and CARTWRIGHT D., **Structural Models: An Introduction To the Theory of Directed Graphs**, J. WILEY & SONS, INC. New York, 1965.
9. HSIEH, F. and CHANG, S., **Dispatching-Driven Deadlock Avoidance Controller Synthesis for Flexible Manufacturing System**, IEEE TRANS. ROBOTICS AND AUTOMATION, Vol. 10, No. 2, April 1994, pp. 196-209.
10. LEUNG, Y.T. and SHEEN, G.J., **Resolving Deadlocks in Flexible Manufacturing Cells**, JOURNAL OF MANUFACTURING SYSTEMS, Vol. 12, No. 4, 1994, pp. 291-304.
11. PEDGEN, C.D., **Introduction to SIMAN**, System Modelling Corporation, State College, PA, USA, 1993.
12. REINGOLD, E.M., NIEVERGELT, J. and DEO, N., **Combinatorial Algorithms**, PRENTICE HALL, Englewood Cliffs, NJ, USA, 1977.
13. VISWANADHAM, N., NARAHARI, Y. and JOHNSON, T.L., **Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models**, IEEE TRANS. ROBOTICS AND AUTOMATION, Vol. 6, No. 6, December 1990, pp. 713-723.
14. WYSK, R.A., YANG, N.S. and JOSHI, S., **Detection of Deadlocks in Flexible Manufacturing Cells**, IEEE TRANS. ROBOTICS AND AUTOMATION, Vol. 7, No. 6, December 1991, pp. 853-859.
15. WYSK, R.A., YANG, N.S. and JOSHI, S., **Resolution of Deadlocks in Flexible Manufacturing Systems: Avoidance and Recovery Approaches**, JOURNAL OF MANUFACTURING SYSTEMS, Vol. 13, No. 2, 1994, pp. 128-138.