2012 IEEE International Conference on Control Applications (CCA)
Part of 2012 IEEE Multi-Conference on Systems and Control
October 3-5, 2012. Dubrovnik, Croatia

Matrix model based control of Flexible Manufacturing Systems using Banker's algorithm

Antonio Krnjak, Tamara Petrovic, Stjepan Bogdan

Abstract— **Manufacturing systems belong to a class of discrete-event systems. Main research objectives here include development of suitable models for manufacturing systems and their analysis. Moreover, various control algorithms are being developed to maximize utilization and throughput, while trying to achieve acceptable level of time and space complexity. In general, there is always a trade off between given objectives. In this paper we propose an exact mathematical procedure to relate matrix model of manufacturing systems and control policy based on Banker's algorithm. Matrix model allows straightforward application in industrial systems as well as structural analysis. On the other side, it gives a framework for application of various versions of Banker's algorithm.**

I. INTRODUCTION

In past twenty years, manufacturing systems have been a challenging subject in research community. Due to increase in complexity of manufacturing procedures, and increasing of computer power, aim is to develop control policy that optimizes resource utilization and production throughput.

Manufacturing systems consist of a set of various resources, such as specialized machines, buffers, robotic manipulators and automated guided vehicles, that need to be handled and controlled to perform a specific production function. Mathematically, manufacturing systems can be studied as Discrete Event Dynamical Systems (DEDS)[1].

The need to control manufacturing system rises from the fact that the number of resources that production operations use is limited, hence resource sharing is required. Control policy needs to take care of prioritized assignment of resources and avoidance of undesired states - deadlocks. If resources were unlimited, and each operation would have its own designated machine, control policy would involve only a simple sequencing of operations.

There are two different approaches to control of manufacturing systems: centralized and distributed approach. Distributed control is oriented towards application in large-scale systems [2]. However, due to safety issues, in industrial applications centralized solutions are still the dominant ones.

General type of DEDS include arbitrarily sequenced operations, with alternative part routings, including assembly or disassembly operations. It is proved that optimal control of such a general systems is not feasible in polynomial time [3]. Therefore, various suboptimal algorithms and optimal algorithms for subclasses of DEDS have been developed. However, due to increase in computer power, for middle-scale manufacturing systems some non-polynomial algorithms can be also applied [4].

There are three main tools when dealing with modeling and control of manufacturing systems: Petri nets [4], graph-theoretical approach [5] and automata [6], [7]. Each of these approaches has been exhaustively studied over the past decades, and a good overview can be found in [8]. Control based on Petri nets focuses on identification of structures called siphons that should be properly supervised [9], [10]. Petri net approach is theoretically very strong, and provides very good simulation results, but still suffers from high complexity issues, especially for large scale systems.

Many control policies ([11],[12]) were inspired by Banker's algorithm for operating systems [13]. Banker's algorithm is a deadlock avoidance strategy that controls resource allocation to ensure that system is always safe, and its complexity is lower then similar Petri nets algorithms.

Papers in the literature mostly do not discuss implementation details, but rather concentrate on mathematical aspect of control policies. In this paper we use matrix model of manufacturing systems with straightforward industrial applicability that incorporates structures for monitoring and system control. Matrix model is in a direct correspondence with Petri nets, and vice-versa [9].

We propose an exact mathematical procedure that transforms a system given in matrix model (or a corresponding Petri net) to obtain matrices needed for Banker's algorithm. This allows straightforward implementation and testing of various algorithms based on Banker's algorithm. The approach is valid for a class of Multiple-Reentrant Flowlines (MRF) with multi-resource operations (in the literature this class is also referred to as Sequential Resource Allocation Systems, S-RAS, with fixed production routs [14]). Proposed procedure can be also applied to more complex systems with suitable decomposition [14].

The paper is organized in the following way. In chapters II and III we describe matrix model and Banker's control algorithm. In chapter IV we describe the mathematical procedure for transformation to matrices suitable for Banker's algorithm, with simulation case study given in chapter V. Results are presented at the end of the paper.

II. MATRIX MODEL OF A MANUFACTURING SYSTEM

Manufacturing systems in general consist of a sequence of operations (tasks) that need to be executed on raw materials and parts to get an end product. Each operation requires one or several resources.

Laboratory for Robotics and Intelligent Control Systems (LARICS), Department of Control and Computer Engineering, University of Zagreb, Unska 3, 10 000 Zagreb, Croatia (tamara.petrovic@fer.hr)

Class of systems that are considered in this paper are MRF systems, with the following properties:

- there are no assembly or choice jobs
- system can have shared resources
- there are no two subsequent operations that require the same resource (no self-loops, with at least a buffer operation in between)
- each part has a clear start and end (input and output).

Given a set of operations and a set of resources that compose a manufacturing system, to obtain a matrix model we describe system activities in form of *if-then* rules. This rules correspond to lingual (verbal) description of a production process. For example, one if-then rule can be: 'if operation 1 is done and resource B is ready for operation 2, then release resource A and start operation 2 using resource B'. Mathematically, each rule corresponds to a component of the binary *logical state vector*, denoted $\mathbf{x}$. A particular rule is fulfilled (corresponding element of vector x is set from '0' to '1') if all preconditions in *if* part of the rule are satisfied. In that case, actions from *then* part are started.

Matrix model is a structured notation of elements included in if-then rules and it consists of eight matrices: $\mathbf{F_u}$, $\mathbf{F_v}$, $\mathbf{F_r}$, $\mathbf{F_y}$, $\mathbf{S_u}$, $\mathbf{S_v}$, $\mathbf{S_r}$ and $\mathbf{S_y}$. Matrices $\mathbf{F_x}$ correspond to *if* part, and matrices $\mathbf{S_x}$ to *then* part of rules.

Thorough description of matrix model can be found in [9]. In the text that follows we will illustrate forming of matrix model on a simple manufacturing systems shown in Fig.1. We will discuss only those matrices that are relevant to the proposed control policy.

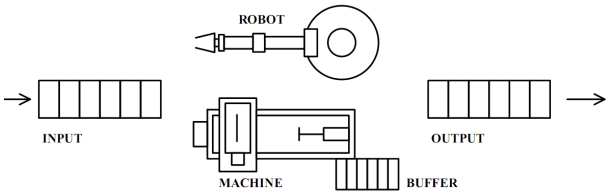

Fig. 1: Manufacturing system

System comprises one robotic arm R, one specialized machine M and a six-slot buffer B. If there is a part on the input conveyor and robotic arm R is idle, then R takes the part to machine M (operation RP1). If part is moved and M is idle, then M starts to process the part (MP) ans R is released. Once machine M is done and there is a place in the buffer, part moves to the buffer (BP). If there is a part in buffer and R is idle, R moves the part from machine M to the output tray (RP2).

We first determine *resource vector* as: $\mathbf{r}$ = [R M B] and *operations vector*: $\mathbf{v}$ = [RP1 MP BP RP2].

In matrices $\mathbf{F_u}$, $\mathbf{F_r}$ and $\mathbf{F_v}$, rows correspond to *if-then* rules, and columns correspond to input places, resources and operations, respectively.

$$\mathbf{F_u} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{F_r} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{F_v} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In matrices $\mathbf{S_r}$, $\mathbf{S_v}$ columns correspond to *if-then* rules and rows to resources and operations, respectively.

$$\mathbf{S_r} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{S_v} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

For example, second rule ($\mathbf{x}(2)$) is: 'if resource M is idle and operation R1P is done, then release resource R and start operation MP'. This rule corresponds to the second row of $\mathbf{F_r}$ and $\mathbf{F_v}$ (if part) and second column of $\mathbf{S_r}$ and $\mathbf{S_v}$ (then part)).

Except for matrices, we define three vectors that represent system feedback: *input vector* $\mathbf{u}$ ($\mathbf{u}(i) = 1$ if there is one part on system entry), *idle resource vector* $\mathbf{r_a}$ ($\mathbf{r_a}(i) = 1$ if one resource i is idle, otherwise, $\mathbf{r_a}(i) = 0$), *completed operations vector* $\mathbf{v_c}$ ($\mathbf{v_c}(j) = 1$ if operation j is completed, otherwise, $\mathbf{v_c}(j) = 0$) and *output vector* $\mathbf{y}$ ($\mathbf{y}(i) = 1$ if there is one part on system output). We define system state vector as $\mathbf{m} = [\mathbf{u}^T \mathbf{v_c}^T \mathbf{r_a}^T \mathbf{y}^T]^T$. If system in Fig.1 is empty: $\mathbf{u}$ = [0], $\mathbf{r_a}$ = $[1\ 1\ 6]^T$ and $\mathbf{v_c}$ = $[0\ 0\ 0\ 0]^T$.

Having this defined, we determine logical state vector as:

$$\bar{\mathbf{x}}(k) = \mathbf{F_v} \Delta \bar{\mathbf{v}}_\mathbf{c}(k-1) \nabla \mathbf{F_r} \Delta \bar{\mathbf{r}}_\mathbf{a}(k-1) \nabla \mathbf{F_u} \Delta \bar{\mathbf{u}}(k-1), \quad (1)$$

where Δ and ∇ denote binary matrix multiplication and addition, and k denotes an iteration step.

Signals for starting of associated operations are calculated according to:

$$\mathbf{v_s} = \mathbf{S_v} \Delta \mathbf{x} \quad (2)$$

$$\mathbf{r_s} = \mathbf{S_r} \Delta \mathbf{x} \quad (3)$$

where $\mathbf{v_s}(j) = 1$ if operation j is going to be started and $\mathbf{r_s}(i) = 1$ if resource i is going to be released. Structure of manufacturing system control that is based on matrix model is shown in Fig. 2.

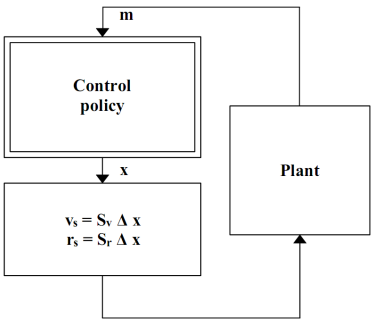

Fig. 2: Structure of a control system

Control policy first determines set of fulfilled rules (equation (1)) and then sets, if necessary, particular elements of vector $\mathbf{x}$ to '0'.

Here should be noted that only a partial information on the system state can be derived on from its direct sensory inputs. In particular, if an operation j is in progress, but is not yet completed, $\mathbf{v_c}(j) = 0$, which as well holds if operation is not in progress. Since controller needs to have the whole information, we will use a *static state vector* $\tilde{\mathbf{m}} = [\tilde{\mathbf{u}}^T \tilde{\mathbf{v}}^T \tilde{\mathbf{r}}^T \tilde{\mathbf{y}}^T]^T$ to memorize the state of operations and resources, regardless of the sensory input. In other words, $\tilde{\mathbf{v}}(j) = 1$ if operation j is in progress or completed, otherwise is '0').

III. Control policy based on Banker's algorithm

The nature of a discrete-event controller is basically similar to every other controller: based on current state of a system and the behavior we want to obtain, controller should autonomously decide which actions to take to achieve the goals, possibly with the smallest amount of energy spent.

In manufacturing systems there are three main issues to consider when designing a controller: i) exact sequencing of operations ii) deadlock avoidance iii) priority assignment in case of a conflict. Sequencing of operations is inherent property of matrix model and is obtained by using equations (1-3).

Deadlock avoidance is the most important aspect of controller design and should be given special attention. For system in Fig.1, deadlock is the following situation: robotic arm R holds a part to release in machine M. In the same time machine M is full and buffer B is full holding six manufactured parts. The system is blocked since robotic arm cannot unload the buffer while holding a part. To resolve such a deadlock situation one needs to manually unload a part from any of the resources involved in deadlock. Therefore, strategy that avoids these kind of situations should be implemented.

Third issue that should be discussed is conflict resolution. Conflict occurs when two or more operations are requesting the same resource. For example, if there is both one part on the input conveyor, and one in the buffer, controller should decide which part robot will take. In general, policies that favor system loading tend to result in higher levels of resource utilization, but are more prone to deadlocks. On the contrary, policies that favor system unloading are less prone to deadlock and suffer from lower utilization.

Control policy described herein is based on Banker's algorithm [13]. The starting point for Banker's algorithm for deadlock avoidance is to define system *resources* and *processes* that are trying to allocate those resources. Information about resources can be directly derived from matrix model vectors $\mathbf{r}$ and $\mathbf{r_a}$.

A single *process* is associated to each part that enters a manufacturing line. Each process is a sequence of operations a part goes through in order to be completely manufactured. For the system depicted in Fig.1, if there is one part (P1) on input conveyor, and one part (P2) in the buffer, then two active processes exist. First process corresponds to part P1 and is going to need all three resources to terminate, and second process corresponds to part P2 and needs only to allocate resource R to finish. It it important to note here that process depends on the system state and will change during production, as the production of part advances. Maximum number of active processes is equal to the number of slots available for part storage.

The information about relation between active processes and resources is depicted in *allocation* matrix $\mathbf{A}$ and *need* matrix $\mathbf{N}$.

Matrix $\mathbf{A}$ shows which resource a particular process currently holds (allocates), while matrix $\mathbf{N}$ shows which resources a particular process needs in order to terminate. In both matrices columns and rows correspond to resources and processes, respectively.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{N} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrices $\mathbf{A}$ and $\mathbf{N}$ depend on system structure and current state and therefore change as the system evolves. Resources are granted to processes sequentially, one by one. Each time a process requests a resource, Banker's algorithm updates matrices $\mathbf{A}$ and $\mathbf{N}$ and checks whether this allocation would result in a *safe* state. State is safe if all active processes can be successfully terminated, without deadlock.

Banker's algorithm checks if, after the requested resource is granted, processes could terminate in a sequential manner, one at a time. In such a scenario, a process is granted all resources to terminate, and afterwards it releases all allocated resources. If all processes could end in such a manner, initial allocation is considered to be safe. This scenario is the worst case scenario, but from the theoretical point of view it is important to guarantee that such an option exists [12]. In a given example, if process P2 requests robotic arm, possible termination sequence is: process P2 terminates first, and afterwards process P1 can grant all requested resources. Hence, control policy would allow allocation of R to P2. Note that, if process P1 would request resource R, Banker's algorithm would likewise grant R to P1. In this case process P1 can terminate first since machine M is idle and there are 5 free slots in the buffer. After P1 terminates and deallocates resource, process P2 can finish.

IV. Deriving Banker's matrices from matrix model

In order to implement presented control policy we propose a mathematical procedure that determines matrices $\mathbf{N}$ and $\mathbf{A}$ dynamically based on current system state.

First, three static matrices ($\mathbf{A_c}$, $\mathbf{N_c}$ and $\mathbf{R_c}$) are determined off-line since they depend solely on system structure. Dynamical matrices $\mathbf{A}$ and $\mathbf{N}$ are then determined by multiplication of system state vector $\tilde{\mathbf{m}}$ and the static matrices. We illustrate the procedure for system in Fig.1, with current state such that there is one part on the input conveyor and one part in buffer B (two active processes) and idle resource vector is $\mathbf{r_a} = [1\ 1\ 5]^T$.

***Definition** 1:* Matrix $\mathbf{A_c}$ relates set of inputs and operations to set of resources. If operation i is performed by resource j then $\mathbf{A_c}(i, j) = 1$, otherwise $\mathbf{A_c}(i, j) = 0$.

Matrix $\mathbf{A_c}$ is determined using the following relation:

$$\mathbf{A_c} = (\mathbf{S_r} \cdot [\ \mathbf{F_u}\ \mathbf{F_v}\ ])^T. \quad (4)$$

For system in Fig.1:

$$\mathbf{A_c} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

First row of matrix $\mathbf{A_c}$ corresponds to input place and does not need a resource. Second row corresponds to operation RP1 and is performed by resource R, thus '1' is placed in the first column.

***Definition** 2:* Matrix $\mathbf{A}$ relates set of input places and operations to set of system resources. If operation i currently holds resource j, $\mathbf{A}(i,j) = 1$, otherwise $\mathbf{A}(i,j) = 0$.

Matrix $\mathbf{A}$ is in iteration step k determined as:

$$\mathbf{A}(k) = diag([\ \tilde{\mathbf{u}}(k)^T\ \tilde{\mathbf{v}}(k)^T\ ]) \cdot \mathbf{A_c}, \quad (5)$$

where *diag*($\mathbf{a}$) forms a diagonal matrix with diagonal elements given in row-vector $\mathbf{a}$.

For system in Fig.1, in iteration step k: $\tilde{\mathbf{u}}(k)$ = [1] and $\tilde{\mathbf{v}}(k) = [0010]^T$ and matrix $\mathbf{A}(k)$ is:

$$\mathbf{A}(k) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

This matrix $\mathbf{A}$ comprises the same information like matrix $\mathbf{A}$ given in chapter III, with one null-row added for each empty slot in the system.

To define procedure for matrix $\mathbf{N}$ we first define static matrices $\mathbf{G_v}$ and $\mathbf{N_c}$ that are calculated off-line:

***Definition** 3:* Matrix $\mathbf{G_v}$ defines precedence relations among operations. If operation j is a direct follower of operation i then $\mathbf{G_v}(i,j) = 1$, otherwise $\mathbf{G_v}(i,j) = 0$.

Matrix $\mathbf{G_v}$ can be calculated as:

$$\mathbf{G_v} = (\mathbf{F_v} \cdot \mathbf{S_v})^T. \quad (6)$$

For system in Fig.1:

$$\mathbf{G_v} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

***Definition** 4:* Matrix $\mathbf{N_c}$ relates set of input places and operations with set of resources. If resource j is once required downstream of operation i in a production line, then $\mathbf{N_c}(i,j) = 1$, otherwise $\mathbf{N_c}(i,j) = 0$. If resource j is required k times downstream of operation i, then $\mathbf{N_c}(i,j) = k$.

Matrix $\mathbf{N_c}$ can be calculated according to:

$$\mathbf{N_c} = \sum_{p=1}^{N_o} [\ \mathbf{F_u}\ \mathbf{F_v}\ ]^T \cdot \mathbf{G_v}^p \cdot \mathbf{S_r}^T \quad (7)$$

where N_o is the number of operations in the longest manufacturing line. If j is p^{th} downstream operation after i, then $\mathbf{G_v}^p(i,j) = 1$, hence, summand in (7) corresponds to the resource that performs operation j.

***Theorem** 1:* For MRF class of manufacturing systems, matrix $\mathbf{N_c}$ can be determined using the following relation:

$$\mathbf{N_c} = [\ \mathbf{F_u}\ \mathbf{F_v}\ ]^T \cdot \mathbf{G_v} \cdot (\mathbf{I} - \mathbf{G_v})^{-1} \cdot \mathbf{S_r}^T \quad (8)$$

Proof: In MRF class of systems each line has a clearly defined beginning and end. Since the longest manufacturing line has N_o operations, it follows that $\forall p > N_o, \mathbf{G_v}^p = \mathbf{0}$. Then:$(I + \mathbf{G_v} + \mathbf{G_v}^2 + \mathbf{G_v}^3 + \dots) = (I - \mathbf{G_v})^{-1}$ [15]. According to (7):

$$\mathbf{N_c} = \sum_{p=1}^{N_o} [\ \mathbf{F_u}\ \mathbf{F_v}\ ]^T \cdot \mathbf{G_v}^p \cdot \mathbf{S_r}^T = \sum_{p=1}^{\infty} [\ \mathbf{F_u}\ \mathbf{F_v}\ ]^T \cdot \mathbf{G_v}^p \cdot \mathbf{S_r}^T =$$

$$= [\ \mathbf{F_u}\ \mathbf{F_v}\ ]^T \cdot \mathbf{G_v}^p \cdot (\sum_{k=0}^{\infty} \mathbf{G_v}^p) \cdot \mathbf{S_r}^T =$$

$$= [\mathbf{F_u}\mathbf{F_v}]^T \cdot \mathbf{G_v} \cdot (I - \mathbf{G_v})^{-1} \cdot \mathbf{S_r}^T.$$

♦

For the system in Fig.1 matrix $\mathbf{N_c}$ is equal to:

$$\mathbf{N_c} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

To finish a part after operation RP1, one needs all system resources, thus second row of the matrix contains no '0' elements. Likewise, part in machine M (third row) needs buffer B and then robot R, hence '0' element is in the second column.

Once matrix $\mathbf{N_c}$ is calculated off-line, we can dynamically determine need matrix $\mathbf{N}$ in iteration step k by:

$$\mathbf{N}(k) = sgn(diag([\ \tilde{\mathbf{u}}(k)^T\ \tilde{\mathbf{v}}(k)^T\ ]) \cdot \mathbf{N_c}), \quad (9)$$

where *sgn* is signum function.

For the given example:

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

We can see that matrix $\mathbf{N}$ depicts only those rows of matrix $\mathbf{N_c}$ that are currently active, that is, which hold a part.

***Definition** 5:* Matrix $\mathbf{R_c}$ relates set of input places and operations with set of resources. If resource j is required for

operation immediately after input/operation i, then $\mathbf{R_c}(i,j) = 1$, otherwise $\mathbf{R_c}(i,j) = 0$.

Matrix $\mathbf{R_c}$ is determined as:

$$\mathbf{R_c} = [\ \mathbf{F_u}\ \mathbf{F_v}\ ]^T \cdot \mathbf{G_v} \cdot \mathbf{S_r}^T. \quad (10)$$

For our example:

$$\mathbf{R_c} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

***Definition** 6:* Matrix $\mathbf{R}$ relates set of input places and operations with set of resources. If operation i in current state requests resource j, then $\mathbf{R}(i,j) = 1$, otherwise $\mathbf{R}(i,j) = 0$.

Matrix $\mathbf{R}$ is determined based on dynamical state vectors $\mathbf{u}$ and $\mathbf{v_c}$ since it is changed only when some operation, that is currently in progress, terminates. We determine the request matrix $\mathbf{R}$ in iteration step k as:

$$\mathbf{R}(k) = sgn(diag([\ \mathbf{u}(k)^T\ \mathbf{v_c}(k)^T\ ] \cdot \mathbf{R_c}) \cdot diag(\mathbf{r_a}(\mathbf{k}))). \quad (11)$$

For the given example, if BP is still in progress, ($\mathbf{u}(k) = [1]$, $\mathbf{v_c}(k)$ = [0 0 0 0]^T, $\mathbf{r_a}(k)$ = [1 1 0]^T), only input part request the robot R:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Banker's algorithm is called if there is a non-zero element in matrix $\mathbf{R}$. Input parameters for the algorithm are matrices $\mathbf{A}$, $\mathbf{N}$ and $\mathbf{R}$ and idle resource vector $\mathbf{r_a}$. Structure of the control policy is given in Fig.3.

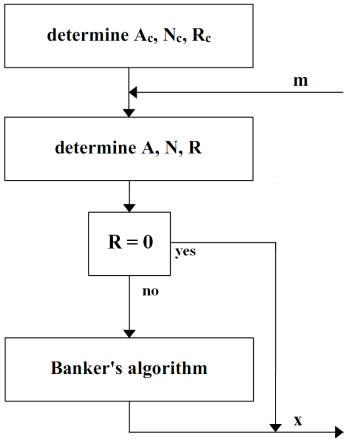

Fig. 3: Control policy diagram

From time complexity point of view, procedure is polynomial in the maximum number of active processes. If we denote with n_u number of inputs, n_v number of operations and n_r number of resources, then dimensions of static matrices are $(n_u + n_v, n_r)$ and static state vector $(n_u + n_v, 1)$. Number of basic calculations for calculation of dynamic matrices is $n_r \cdot (n_u + n_v)^2$ (matrix multiplication). Time complexity of Banker's algorithm is in the worst case $O((n_u + n_v)^2)$. Since all given matrices are sparse, time and space complexity can be reduced by using structures and algorithms specially modified for spare matrices [16].

V. Case Study

The proposed control policy, based on Banker's algorithm, has been tested on manufacturing system shown in Fig.4.

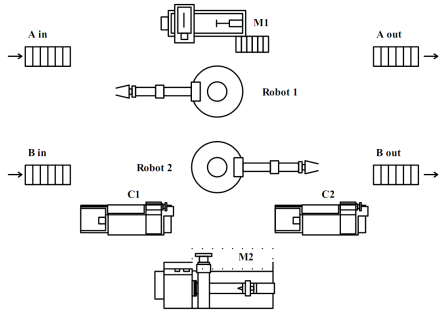

Fig. 4: Case study example

System consists of five resources and manufactures two different part types: A and B. Part A: robot R1 moves part A from the input conveyor to machine M1 and afterwards to the output conveyor. Part B has a more complex manufacturing procedure: first robot R1 moves raw part B to machine C1, then robot R2 moves it to machine M2 and from machine M2. Last, robot R1 moves part B from machine C2 to the output conveyor. To sum up, resource sequence for part A is: R1 - M1 - R1, and for part B: R1 - C1 - R2 - M2 - R2 - C2 - R1. All resources can hold maximally one part at a time.

This is an example of a complex system with structural properties that can lead to higher level deadlocks ([17]) and is difficult to control. We first define resource vector as: $\mathbf{r}$ = [R1 M1 C1 R2 M2 C2] and operations vector as: $\mathbf{v}$ = [R1 M1 C1 R2 M2 C2]. Initially system has five raw parts at each input conveyor: $\mathbf{u}$ = [5 5]^T, and all resources are idle: $\mathbf{r_a}$ = [1 1 1 1 1 1]^T and $\mathbf{v_c}$ = [0 0 0 0 0 0 0 0 0 0 0 0]^T. Static matrices $\mathbf{A_c}$, $\mathbf{N_c}$ and $\mathbf{R_c}$ for the case study system are:

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T$$

$$N_c = \begin{bmatrix} 2 & 2 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}^T$$

$$R_c = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$$

Dynamic matrices **N** and **R** for the case study system in the initial state are:

$$N = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$R = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

In the initial state only input places hold parts, therefor only first two rows of matrices **N** and **R** are non-zero. Matrix **A** is a zero-matrix since all resources are initially idle.

Control policy based on Banker's algorithm was applied. Simulations results depicted in Fig. 5 show that system operates without deadlocks, and after 57 time units ten end-products get out of the system. Conflict resolution strategy is chosen to prefer part A over part B and to favor system loading. Since part A production is shorter and of higher priority, all parts A are produced first and then parts B. Conflict resolution strategy can be changed to use some other priority rule.

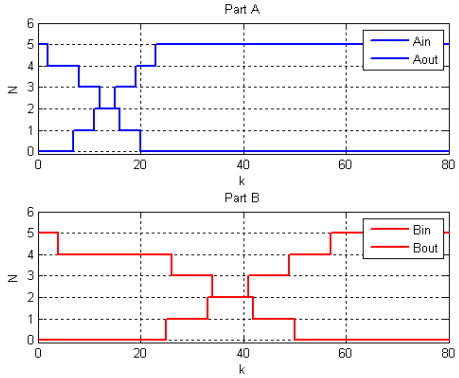

Fig. 5: Control policy based on Banker's algorithm

VI. Conclusion

We presented a generic procedure for deadlock-free control of manufacturing systems using matrix model and Banker's algorithm. Procedure is simple and straightforward and consists of dynamical determination of system matrices used in Banker's algorithm. We implemented the basic version of Banker's algorithm but more complex algorithms can be likewise used.

Results of the case study simulation show that system works properly without deadlocks. Although for given system there were no problems considering time complexity, for larger scale systems one should consider using appropriate memory management system and algorithms suitable for sparse matrices.

In the future, we plan to modify this procedure to be suitable for class of Free-choice Multiple Reentrant Flowlines (FMRF), which contain choice-jobs, non-fixed part routing and assembly operations. Likewise, procedure could be modified to use as a starting point some other formalisms, such as Petri Nets, instead of a matrix model.

ACKNOWLEDGEMENT

This work has been supported by the European Commission FP7-ICT-2011-7 project "Estimation and Control for Safe Wireless High Mobility Cooperative Industrial Systems" (EC-SAFEMOBIL, Project No. 288082).

References

[1] C. G. Cassandras, S. Lafortune, "Introduction to Discrete Event Systems",Springer-Verlag New York, Inc.,2006.
[2] M.V.Iordache, P.J.Antskalis: "Decentralized supervision of Petri Nets", IEEE Transactions on Automatic Control, Vol.51, No.2, 2006.
[3] E. Roszkowska, S. Reveliotis: "Establishing the NP-hardness of maximally permissive RAS-based approaches to multi-vehicle system safety", ICRA 2010, Anchorage, pp. 322-327
[4] N. Q. Wu, M. C. Zhou, "Shortest Routing of Bidirectional Automated Guided Vehicles Avoiding Deadlock and Blocking", IEEE/ASME Trans.on Mechatr., Vol. 12, 1, 2007, pp. 63-72
[5] M. P. Fanti, B. Turchianod, "Deadlock Avoidance in Automated Guided Vehicle Systems", Proc of 2001 IEEE/ASME Int'l Conf on Advanced Intelligent Mechatronics, Como, 2001, pp. 1017-1022
[6] P.J.Ramadge, W.M.Wonham: "The control of discrete event systems", Proc. IEEE, vol. 77, no. 1, pp.81-89, Jan. 1989.
[7] S. A. Reveliotis, "Conflict resolution in AGV systems," IIE Trans, vol.32, no. 7, 2000, pp. 647-659
[8] Z. Li, N. Wu, M. Zhou: "Deadlock Control of Automated Manufacturing Systems Based on Petri Nets - A literature review", IEEE. Transactions on Sytems, Man. and Cybernetics, 2011.
[9] S. Bogdan, F.L.Lewis, Z.Kovacic, J.Mireles Jr., "Manufacturing Systems Control Design", Springer, 2006.
[10] S. Wang, M. Zhou, C. Wang: "Extracting all minimal siphons from maximal unmarked siphons in manufacturing-oriented Petri nets", CASE 2011, Trieste, Italy, pp. 399-404
[11] J.Ezpeleta, F. Tricas, F. Garcia-Valles, J.M.Colom: "A Banker's Solution for Deadlock Avoidance in FMS With Flexible Routing and Multiresource States", IEEE Transactions on robotics and automation, col. 18, No.4, 2002.
[12] L. Kalinovcic, T.Petrovic, S.Bogdan, T.Petrovic: "Modified Banker's algorithm for scheduling in multi-AGV systems", CASE 2011, Trieste.
[13] E.W.Dijkstra: "The mathematics behind the Banker's Algorithm", Selected Writings on Computing: A Personal Perspective, Springer-Verlag, 1982.
[14] J.Ezpeleta, L.Recalde: "A deadlock avoidance approach for Non-Sequential Resource Allocation Systems", IEEE. Transactions on Sytems, Man. and Cybernetics, 2002.
[15] J.N.Franklin: "Matrix Theory", Prentice-Hall, 1968.
[16] Z. Zlatev: "Computational Methods for General Sparse Matrices (Mathematics and Its Applications)", Springer, 1991.
[17] S. Lee, D.M. Tilbury: "Deadlock-Free resource Allocation Control for a reconfigurable Manufacturing System with Serial and Parallel Configuration", IEEE Transactions on Systems, Man. and Cybernetics, Vol.37, No.6, 2007.