European Journal of Operational Research 57 (1992) 287–304
North-Holland

Performance evaluation of cellular flexible manufacturing systems: A decomposition approach

R. Ram and N. Viswanadham

Department of Computer Science and Automation, Indian Institute of Science, Bangalore-560012, India

Received June 1990; revised December 1990.

Abstract: In a cellular flexible manufacturing system, a majority of the part types manufactured by the system require machining in a single cell only. However, a few part types (called *rare* parts) need processing in two or more cells. It is precisely the intercell movement of these rare parts that leads to weak interaction among the otherwise independent cells. We show how this structure can be exploited to achieve significant computational savings in the performance evaluatiion of the cellular flexible manufacturing systems by using product form closed queueing network models and generalised stochastic Petri net models. In product form closed queueing network models, the performance measures of individual cells are computed first, and then the system performance measures are obtained from the performance measures of the cells, suitably accounting for cell interactions. In generalised stochastic Petri net models, the structure of the system enables us to apply the techniques of near-complete decomposability to the the underlying continuous-time Markov chain. The state aggregation in the Markov chain is based on identifying the fast and slow transitions corresponding to intra-cell and inter-cell workpiece movements, respectively. This decomposition leads to a significant reduction in the state space of the Markov chain to be analysed. We also present a comparison of the computational requirements of the analysis using decomposition approach and the exact analysis.

Keywords: Cellular manufacturing, flexible manufacturing, performance evaluation, decomposition methods

1. Introduction

A Flexible Manufacturing System (FMS) is a highly automated manufacturing system well suited for the simultaneous production of a wide variety of part types in low- to mid-volume quantities at a low cost while maintaining a high quality on the finished products. An approach to the implementation of FMSs, which has attracted much attention and has gained wide acceptability, is the technique of Group Technology (GT), [5,11,26], which leads to configuring FMSs as Cellular FMSs (CFMSs), or simply Cellular Manufacturing Systems (CMSs). Group technology focusses on cutting down the proportion of time spent on non-productive activities like material handling and waiting, relative to the actual machining time of a part. This is achieved by organising the manufacturing system as follows. Part types which have similar processing requirements are identified and grouped together into *part families.* The part types in each familiy are characterised by similarities in design and manufacture. This similarity is translated into manufacturing efficiency by arranging the workcentres required for production of the part family into a group of machines called a *cell* to facilitate smooth and efficient work flow.

In this paper, we formulate and solve the performance evaluation problem of a CMS under the following assumptions:

1. The various part types manufactured by the

CMS belong to exactly one of two categories of part families: *local and rare.* A *local* part family is characterised by the fact that each of its member part types (called a *local part*) needs to undergo processing only on the machines in the corresponding machine cell. In contrast, a *rare* part family is one whose members (called *rare* parts) have to undergo machining on workcentres located in more than one cell. Ideally, in a CMS there should be no rare parts; however, due to the constraints in the available number of certain machine centres, inter-cell moves of the rare parts become unavoidable since it may not be possible to dedicate all the machines required by each part family [14].

2. For each machine (respectively, cell), the work content (defined, for an individual machine, as the product of the number of workpieces and the machining time per workpiece; for a cell, it is the sum of the workcontents of each part at each machine in the cell where it is processed) of a rare part type (respectively, rare part family) using that machine (respectively, cell) is far smaller than the work content of a local part type (respectively, local part family) processed at that machine (respectively, cell).

3. Within a cell, a part type, whether local or rare, undergoes processing on almost all of its machine centres.

4. Owing to the overwhelming majority of local-type workpieces over rare-type workpieces, within cell (i.e., intra-cell) part movements are far more frequent than inter-cell part moves out of a cell. The interaction among various cells is caused only due to the inter-cell movement of the rare part types; otherwise, the system can be treated as a collection of isolated, completely independent cells.

We consider *closed* models of the CMS, i.e., the number of workpieces of each part type is fixed (this number is called the *population* of the part type). This assumption models the availability of a limited number of fixtures for each part type. The processing time of part types is assumed to be exponentially distributed. Each part type is routed *probabilistically* among the machines it requires for processing. The routing of a part type is given by an irreducible, discrete state Markov chain whose states are the machines used by the part type. We are interested in obtaining the steady-state average values of the performance measures of *throughput*, *utilisation*, *queue length*, and *waiting time* for each part type at each machine where it is processed. Material handling is not considered explicitly; a material handling equipment can be modelled as 'machine', with the processing time representing the actual part move time.

The rest of the paper is devoted to developing efficient performance evaluation methodologies for CMSs using Product-Form Closed Queueing Network (PF-CQN) models and Generalised Stochastic Petri Net (GSPN) models. The analysis is based on the near-complete decomposability of the system. The CMS is separated into individual cells, which are analysed individually. The performance measures of individual cells are merged, suitably accounting for the interaction due to rare part movement, to give the system performance measures. It is shown that analysing individual cells one by one and then combining their results, rather than analysing the entire CMS in one step, offers significant computational savings, with little loss of accuracy. The results produced by the PF-CQN analysis are exact, within the scope of the product form assumptions; the results of the GSPN analysis are an excellent approximation, with almost negligible error. We illustrate the methodologies, and highlight the computational savings obtained by the decomposition approach, with an example.

2. Notation

2.1. Part families and machines cells

Consider a CMS consisting of M machines, which are grouped together into C different cells, and manufacturing K different part types. Let $\mathcal{M}_c$ be the set of machines in the cell c, $1 \leqslant c \leqslant C$; let $\mathcal{L}_c$ denote the local part family to which cell c is assigned, and $\mathcal{R}_c$ denote the set of rare parts that undergo processing in cell c. We assume that for each cell c, $\mathcal{R}_c \neq \emptyset$; otherwise, the cell c with $\mathcal{R}_c = \emptyset$ is obviously independent of the rest of the system, and can be exactly analysed in isolation. Let $\mathcal{R} = \bigcup_{c=1}^{C} \mathcal{R}_c$ be the set of rare parts and $\mathcal{L} = \bigcup_{c=1}^{C} \mathcal{L}_c$ be the set of local parts. From the discussion on local and rare parts in the previous

section, the following relations must be satisfied by the sets of local and rare parts:

- The set of rare and local parts together must form the set of part types manufactured in the CMS: $\mathscr{L} \cup \mathscr{R} = \{1, 2, \ldots, K\}$.
- Each of the local part families is disjoint with the set of rare parts: $\mathscr{R} \cap \mathscr{L}_c = \emptyset$ for each cell c, $1 \leqslant c \leqslant C$.
- The local part families corresponding to any two different cells c_1 and c_2 do not have any common part types: $\mathscr{L}_{c_1} \cap \mathscr{L}_{c_2} = \emptyset$, for $c_1 \neq c_2$, i.e., the local part families are pairwise disjoint.

2.2. Population vectors

Let $N_k \geqslant 1$ be the population (number of workpieces in the closed model) of part type k, $1 \leqslant k \leqslant K$. The K-dimensional vector $N = (N_1, N_2, \ldots, N_K)$ is called the *population vector* of the CMS. For the cell c, let $\mathscr{N}_c$ be the $(|\mathscr{L}_c| + |\mathscr{R}_c|)$-dimensional population vector for those parts that undergo machining in this cell. We can write the population vector $\mathscr{N}_c$ of a cell c as the concatenation of two row vectors $\mathscr{N}_{c,l}$ and $\mathscr{N}_{c,r}$, where $\mathscr{N}_{c,l}$ is the $|\mathscr{L}_c|$-dimensional row vector whose components are the populations of the local parts of cell c, and $\mathscr{N}_{c,r}$ is an $|\mathscr{R}_c|$-dimensional row vector whose components are the populations of the rare parts visiting cell c.

2.3. Part routing

Let

- $\mathscr{X}_j$ denote the set of machines on which part j $(1 \leqslant j \leqslant K)$ undergoes processing,
- $\mathscr{Y}_i$ denote the set of part types that require machine i $(1 \leqslant i \leqslant M)$.

The probabilistic routing of part j among the machines in $\mathscr{X}_j$ is given by an irreducible, discrete-time Markov chain whose states represent the machines contained in $\mathscr{X}_j$. If $p^j_{i_1,i_2}$ denotes the probability that a workpiece of part type j moves to machine $i_2 \in \mathscr{X}_j$ after completion of processing on machine $i_1 \in \mathscr{X}_j$, then the transition probability matrix of the routing Markov chain is an $|\mathscr{X}_j|$-by-$|\mathscr{X}_j|$ stochastic matrix with the $p^j_{i_1,i_2}$'s as entries. Let the $|\mathscr{X}_j|$-dimensional vector $\boldsymbol{v}_j$ be the steady-state probability vector of this Markov chain; the value $\boldsymbol{v}_{j,i}$ where $i \in \mathscr{X}_j$ is called the *visit ratio* of part j at machine i.

Let $\tau_{i,j}$ denote the mean of the exponentially distributed *processing time* of part j at machine i. The *traffic intensity* of part j at machine i is defined as $\rho_{i,j} = \boldsymbol{v}_{j,i} \times \tau_{i,j}$.

3. Decomposition of closed product form queueing network models of CMSs

Product form queueing networks [3] are a special class of queueing networks which can elegantly model system features like contention for resources, waiting, simultaneous presence of multiple part types and probabilistic routing, and are analytically tractable. They have been successfully used for performance modelling of FMSs, [7,22] despite the fact that certain simplifying assumptions (called *product form assumptions*) have to be made. A PF-CQN can be algebraically solved for system performance measures by means of computational algorithms like the *Convolution Algorithm* developed by Buzen [6] and Reiser and Kobayashi [20], the *Mean Value Analysis* (MVA) algorithm [21] of Reiser and Lavenberg, or the *Recursion by Chain Algorithm* (RECAL) [8] of Conway and Georganas. In this section, we develop efficient performance evaluation methodologies for PF-CQN models of CMSs using the convolution and MVA algorithms. We assume that each machine works under a First-Come-First-Served (FCFS) policy. Under the product form assumptions, it is required that at a particular machine (say, i), the different part types must have identical, exponentially distributed processing times, i.e., $\tau_{i,j}$ is the same for all part types j processed at machine i; let $\tau_{i,j} = \tau_i$.

3.1. Convolution algorithm

The convolution algorithm [6,20] is based on the computation of an array of elements called *normalisation constants* of the PF-CQN model of the CMS. If $\boldsymbol{G}$ denotes the normalisation constant, the array index is a K-dimensional vector whose range is given by $\boldsymbol{0} \leqslant \boldsymbol{n} \leqslant \boldsymbol{N}$, i.e., $0 \leqslant n_j \leqslant N_j$ for each part type j. The performance measures are determined in terms of the computed values in the $\boldsymbol{G}$-array. The entry $\boldsymbol{G}(\boldsymbol{n})$ denotes the normalisation constant when the population vector is $\boldsymbol{n}$; $\boldsymbol{G}(\boldsymbol{N})$ gives the normalisation constant of the system. We summarize below those aspects of the

unmodified (i.e., sequential) version of the convolution algorithm necessary for applying the efficient *Tree Convolution Algorithm* [16] to CMSs.

3.1.1. Unmodified convolution algorithm

The normalisation constant $G(N)$ is computed by the convolution algorithm from the relation [16]:

$$G(N) = (p_1 \otimes p_2 \otimes \cdots \otimes p_M)(N)$$

where $p_i(\boldsymbol{n})$ is the normalisation constant of the system consisting of machine i alone, with population vector $\boldsymbol{n}$, and $\otimes$ denotes the convolution operator.

The G-value of the single-machine system of machine i is given by [3]:

$$p_i(\boldsymbol{n}_i) = \left[\prod_{h=1}^{h=n_i} \frac{1}{\mu_i(h)}\right] \times n_i! \times \prod_{j=1}^{j=K} \frac{\rho_{i,j}^{n_{i,j}}}{n_{i,j}!}$$

where $n_i = \sum_{j=1}^{K} n_{i,j}$, ($n_{i,j}$ being the number of type-j workpieces at machine i) and $\mu_i(h)$ is the service rate at machine i when a total of h workpieces are waiting/being served at machine i.

The convolution operator is associative and commutative. If a nonempty subset $\mathscr{D}$ of the set of machines is partitioned into two nonempty subsets $\mathscr{A}$ and $\mathscr{B}$, then the g terms of the subnet $\mathscr{D}$ can be evaluated from those of $\mathscr{A}$ and $\mathscr{B}$, using the definition of the convolution operator, as [3]:

$$g_{\mathscr{D}}(\boldsymbol{n}) = \sum_{\boldsymbol{n}_1=\boldsymbol{0}}^{\boldsymbol{n}} g_{\mathscr{A}}(\boldsymbol{n}_1) \Sigma g_{\mathscr{B}}(\boldsymbol{n}-\boldsymbol{n}_1). \tag{1}$$

The sequential convolution algorithm [6,20] performs the convolution operations according to the order of the serial numbers of the machines, by the relation

$$G(N) = (\cdots((p_1 \otimes p_2) \otimes p_3) \otimes \cdots \otimes p_M)(N). \tag{2}$$

The computational requirements for computing the value of G(N) are [20]: (i) *storage requirement:* $\prod_{j=1}^{K}(N_j+1)$, treating the storage for a floating point number as one unit; (ii) *operations:* $M \times K \times \prod_{j=1}^{K}(N_j+1)$ operations, considering the combination of a floating point multiplication and a floating point addition as one operation. Clearly, these values are exponential in the number of part types. The computational requirements are excessive even for moderately sized systems. These requirements arise since the convolution algorithm allows for the possibility that each part type may use all of the machines in the system.

3.1.2. Tree convolution algorithm: Basic concepts

Since the computation of the system normalisation constant $G(N)$ is the key to solving the CQN model by the convolution algorithm, we shall concentrate on methods for its efficient computation. As the number of local-type workpieces is far higher than the number of rare-type workpieces, a typical workpiece is very likely to be of local type and will be processed only at those machines belonging to its corresponding cell (and does not require processing on the rest of the machines in the system), i.e., we have a *sparsity* in routing. Conversely, at each machine only a subset of the total number of part types produced by the CMS undergo processing (i.e., those belonging to the local part family, and the set of rare parts using the cell in which the machine is located). As inter-cell part routing (of rare parts, to machines in other cells) is far less frequent than intra-cell part movement, we also have a *locality* in part routing, meaning that part routing is most frequently confined to an easily distinguishable subset of machines. These two features are used to reduce the computational requirements of computing the normalisation constant G. The tree convolution algorithm [16] developed by Lam and Lien for analysing PF-CQNs with many sparse, localised routing chains provides a formal framework for exploiting the structure of the CMS to efficiently compute the normalisation constant. The concept of decomposition of product form queueing networks has been recently generalised by Conway and Georganas [9].

Consider an arbitrary nonempty subset $\mathscr{A}$ of the set of machines and an arbitrary part type j ($1 \leqslant j \leqslant K$). The usage of the set of machines $\mathscr{A}$ by part j is classified into one of the following categories (we adopt the terminology of [16]):

- If $\mathscr{A} \cap \mathscr{V}_j = \emptyset$, that is, part j does not require processing on any machine in the set $\mathscr{A}$, we say that part j is *noncovered* by $\mathscr{A}$. For example, parts not undergoing machining on any of the machines of cell c are noncovered by $\mathscr{M}_c$. The traffic intensity terms for part j with respect to

the machines $i \in \mathcal{M}_c$ will be zero. It is not necessary to compute G-terms for cell c for those values of the population vector (say, $\boldsymbol{n}_c$) where the number of a noncovered part type is other than zero (i.e., $\boldsymbol{n}_{c,j} > 0$). Hence, we need to compute $G_{\mathcal{M}_c}(\cdot)$-terms only over the index range $\mathbf{0} \leqslant \boldsymbol{n}_c \leqslant \mathcal{N}_c$. This reduces the computational requirements for the evaluation of $G(\cdot)$ terms of cell c by a factor of $\prod_{j=1, j \notin (\mathcal{L}_c \cup \mathcal{R}_c)}^{K} (N_j + 1)$.

- When $\mathcal{L}_j \subseteq \mathcal{A}$, i.e., all the processing needed for manufacturing part j can be accomplished within the set of machines $\mathcal{A}$, part j is said to be *fully covered* by $\mathcal{A}$. For example, all the part types that belong to the local part family of a cell c (i.e., $j \in \mathcal{L}_c$) are fully covered by $\mathcal{M}_c$. In later stages of the convolution, involving machines *not* present in cell c, say $i_1 \notin \mathcal{M}_c$, the corresponding traffic intensity term $\rho_{i_1,j}$ will be zero. Hence, it is sufficient to retain, for future computations, those terms in $G_{\mathcal{M}_c}$ with the population of all the fully covered parts equal to their respective system population, i.e., if $\boldsymbol{n}_c = [\boldsymbol{n}_{c,l} \mid \boldsymbol{n}_{c,r}]$, then it is enough to retain those $G_{\mathcal{M}_c}(\boldsymbol{n}_c)$'s where $\boldsymbol{n}_{c,l} = \boldsymbol{N}_{c,l}$. This represents computational savings by a factor of $\prod_{j \in \mathcal{L}_c} (N_j + 1)$ in future steps involving cell c.
- If $\mathcal{A} \cap \mathcal{L}_j \neq \emptyset$ and $\mathcal{L}_j \not\subseteq \mathcal{A}$, i.e., part j requires processing on some machines in the set $\mathcal{A}$ and also on at least one machine not contained in $\mathcal{A}$, we say that part j is *partially covered* by $\mathcal{A}$. For example, the rare parts visiting cell c are partially covered by the set $\mathcal{M}_c$. For such part types, we need to compute and retain G-terms over their full population range, i.e., over $\mathbf{0} \leqslant \boldsymbol{n}_c \leqslant \boldsymbol{N}_{c,r}$.

In a general PF-CQN, no particular pattern is assumed in the nature of routing. Hence, it is necessary to apply a (heuristic) procedure called *tree planting* [16] to determine any locality and sparsity in routing and rearrange the order of convolution to exploit these for an efficient computation. The tree planting procedure builds a binary tree with each machine centre as a leaf node. Each internal node corresponds to the subsystem consisting of all those machines that are leaf nodes of the subtree rooted at that node; the root corresponds to the entire system. After initialising the G-terms at the leaf nodes (systems of individual machines), a post-order traversal of the tree is carried out starting from the root. At each internal node, the G-arrays corresponding to the left and right children are convolved to obtain the G-value of the subnet formed by the two children. The root node will give the G-values of the entire system. The tree planting step is crucial since it attempts to determine an ordering of computations, using the locality and sparsity in part routing, that minimises the overall computational requirements for calculating $G(\boldsymbol{N})$.

3.1.3. Application of the tree convolution algorithm to CMSs

In the CMS case, we already have a classification of parts into local and rare types, based on their routing, which is directly applicable in the tree planting procedure. If part routing within a subset of nodes shows no locality or sparsity, as in the case of part routing within a cell, it is known [16] that the tree convolution does not offer any advantage over sequential convolution. Hence the $G(\cdot)$-terms of cells can be evaluated by sequential convolution. No tree planting procedure is required for individual cells. If we first compute the $G(\cdot)$-terms for each of the cells individually, and then combine these G-terms, we can obtain significant computational savings: the computation of the $G(\cdot)$-terms of a cell will involve only those parts that use the cell. The convolution of the G terms of two cells (or two disjoint subsets of cells) will involve only those rare parts using either.

The computations for the $G(\boldsymbol{N})$-value may be reordered as

$$G(\boldsymbol{N}) = (G_1 \otimes G_2 \otimes \cdots \otimes G_C)(\boldsymbol{N}) \tag{3}$$

where the term $G_c(\boldsymbol{N})$ denoting the normalisation constant of the subsystem corresponding to cell c is evaluated using

$$G_c(\boldsymbol{N}) = p_{m_1^c} \otimes p_{m_2^c} \otimes \cdots \otimes p_{m_{|\mathcal{M}_c|}^c} \tag{4}$$

where the set of machines in cell c is $\mathcal{M}_c = \{m_1^c, m_2^c, \ldots, m_{|\mathcal{M}_c|}^c\}$.

However, the process of combining the $g(\cdot)$-terms of individual cells to obtain the system normalisation constant can be rendered efficient by the tree convolution algorithm (heuristic methods of tree planting are discussed in [16]). We treat the cells as leaf nodes, and consider only rare part types. Consider the convolution of the G-terms of two nonempty, disjoint subnets $\mathcal{A}$ and $\mathcal{B}$, producing a new subnet $\mathcal{D}$ as their union; $\mathcal{A}$ and $\mathcal{B}$ may each represent one or more (disjoint) subsets of cells. Assume that in

each subnet we retain only those G-terms indexed by the corresponding partially covered part types. The calculation of $G_{\mathscr{D}}$ terms will involve population indices of (only) those part types that are partially covered by either $\mathscr{A}$ or $\mathscr{B}$, i.e., we will compute $G_{\mathscr{D}}(\boldsymbol{n}_{\mathscr{D}})$ terms over $\mathbf{0} \leqslant \boldsymbol{n}_{\mathscr{D}} \leqslant \boldsymbol{N}_{\mathscr{D}}$, where $\boldsymbol{n}_{\mathscr{D}}$ has an index for each $j \in \sigma_{\mathscr{A}} \cup \sigma_{\mathscr{B}}$, where $\sigma_{\mathscr{A}}$ denotes the set of parts partially covered by subnet $\mathscr{A}$.

- If a part type j is partially covered by both $\mathscr{A}$ and $\mathscr{B}$, but is fully covered by $\mathscr{D}$, then we need to retain $G_{\mathscr{D}}$ terms only for index value $n_{\mathscr{D},j} = N_j$, and this information may be implicitly represented in future computations. This gives a computational saving by a factor of N_j in future computations involving $\mathscr{D}$.
- If a part type is partially covered by $\mathscr{D}$, we have to compute and retain terms over its full population range. Parts noncovered by $\mathscr{D}$ have an implicit index of 0.
- The effect of part types fully covered by either of these subnets is already summarised in their respective G-terms.

The tree convolution algorithm for cellular FMSs for the computation of $G(N)$ may be summarised as follows:

1. *Tree planting.* Considering only the set of rare parts, and treating the cells as leaf nodes, determine (heuristically) the binary tree which minimises the computational requirements.

2. *Tree traversal.* Starting from the root node of the binary tree, carry out a post-order traversal.

- If the node is a leaf node, it corresponds to some cell c, $1 \leqslant c \leqslant C$. Compute the G-terms for the cell using sequential convolution, and retain the terms over the full population range of rare parts $\mathscr{R}_c$, corresponding to the full population of the local parts $\mathscr{L}_c$.
- If the node is an internal node, merge the G-terms of the two children by the convolution operator, as explained in the previous paragraphs; we retain G-terms only for those part types partially covered by this internal node.

3. The root node will give the normalisation constant $G(N)$ for the entire system.

3.1.4. Example

The application of tree convolution to CMSs is illustrated through an example. Consider a cellular manufacturing system containing three cells

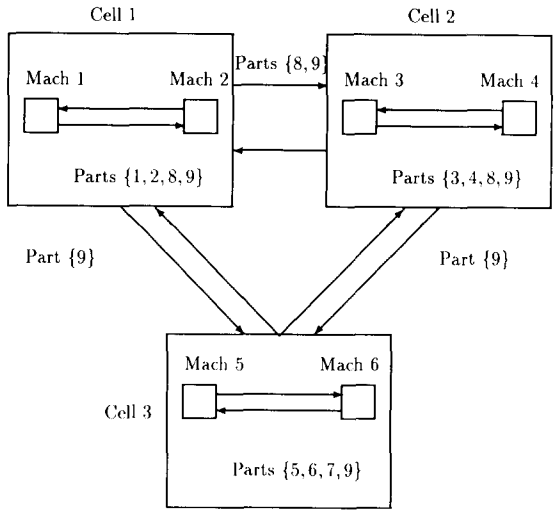

Figure 1. Example cellular FMS

(Figure 1). Each cell contains two machines. The system manufactures nine part types. The details about the machines, local parts and rare parts of each cell are given in Table 1. Table 2 gives the population of each part type, Table 3 gives the mean processing time at each machine and Table 4 gives the visit ratios of each part type at each of the machines on which it is processed. The data used in this example is of illustrative nature.

With the given populations, the space requirement of the sequential algorithm is 15.52×10^6 and the operations count is 838.3×10^6, for the evaluation of $G(N)$. Let us consider the tree convolution algorithm.

1. *Tree planting.* Treating the three cells as leaf nodes, and considering the rare part types 8 and 9 only, we determine a binary tree that minimises the computational requirements during the merging of the G-terms of the cells. In our example, part 8 is fully covered by cells 1 and 2; hence cells 1 and 2 are merged first, and then the subnet {1,2} of cells is merged with the subnet {3}, consisting of cell 3. The corresponding binary

Table 1
Description of the example cellular FMS

Cell, c	$\mathscr{M}_c$	$\mathscr{L}_c$	$\mathscr{R}_c$
1	{1, 2}	{1, 2}	{8, 9}
2	{3, 4}	{3, 4}	{8, 9}
3	{5, 6}	{5, 6, 7}	{9}

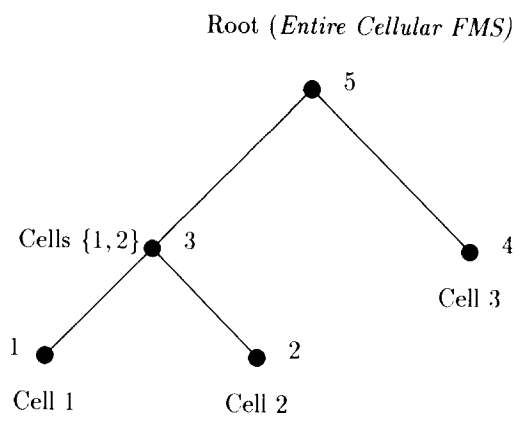

Figure 2. The binary tree used by the tree CQN and tree MVA algorithms

tree is shown in Figure 2. The number beside each node gives the sequence number with which that node is considered for computations during post-order traversal.

2. *Tree traversal.* Starting from the root of the binary tree, we carry out a post-order traversal.

Computations at a leaf node. Consider the node labeled 1, representing the cell 1. Local parts 1 and 2 are fully covered, and rare parts 8 and 9 are partially covered. We compute, using the sequential algorithm, the G-terms of cell 1 over the corresponding population range, viz., $G_1(n_1, n_2, n_8, n_9)$, where $0 \leqslant n_1 \leqslant N_1(=10)$, $0 \leqslant n_2 \leqslant N_2(=10)$, $0 \leqslant n_8 \leqslant N_8(=2)$, $0 \leqslant n_9 \leqslant N_8(=1)$. The indices for parts 3 through 7, which do not undergo processing on any of the machines in cell 1, are implicitly zero in the $G_1(\cdot)$-array. The computation of the $G_1(\cdot)$ needs $\prod_{j \in \mathscr{L}_1 \cup \mathscr{R}_1}(N_j + 1)$ storage units, i.e., 594 storage locations; the operations count is 4752. After computing the $G_1(\cdot)$-array, we need to retain only the portion for the population range of the rare parts 8 and 9, with the index of the local parts 1 and 2 equal to their system population, i.e., we retain $G_1(\cdot)$ terms corresponding to $n_1 = N_1$ and $n_2 = N_2$ over the range $0 \leqslant n_8 \leqslant N_8$, $0 \leqslant n_9 \leqslant N_9$. Hence, the

Table 2
Populations of the parts

	Part								
	1	2	3	4	5	6	7	8	9
Population	10	8	10	10	5	5	5	2	1

Table 3
Mean processing times

	Machine, i					
	1	2	3	4	5	6
τ_i	20.0	10.0	15.0	8.0	20.0	25.0

Table 4
Visit ratios

Part type	Machine					
	1	2	3	4	5	6
1	0.4	0.6	–	–	–	–
2	0.55	0.45	–	–	–	–
3	–	–	0.3	0.7	–	–
4	–	–	0.65	0.35	–	–
5	–	–	–	–	0.5	0.5
6	–	–	–	–	0.5	0.5
7	–	–	–	–	0.6	0.4
8	0.25	0.25	0.25	0.25	–	–
9	0.1	0.1	0.1	0.1	0.2	0.4

Table 5
Tree CQN method: Machine utilisation for each part type

Part type	Machine					
	1	2	3	4	5	6
1	0.5082	0.3811	–	–	–	–
2	0.4253	0.1739	–	–	–	–
3	–	–	0.4392	0.5466	–	–
4	–	–	0.5108	0.1467	–	–
5	–	–	–	–	0.2599	0.3249
6	–	–	–	–	0.2599	0.3249
7	–	–	–	–	0.3374	0.2812
8	0.0566	0.0282	0.0424	0.0226	–	–
9	0.00992	0.00496	0.00744	0.00396	0.0198	0.0496

portion of the $G_1(\cdot)$ to be retained for further computations requires only 6 storage units. Analogous computations are carried out at node 2 (for cell 2) and at node 4 (for cell 3).

Computations at an internal node. Consider node 3, (say, subnet $\mathscr{D}$) of the binary tree, corresponding to merging the G-terms of cell 1 and cell 2; node 3 represents the union {1,2,3,4} of the two disjoint subsets of machines, {1,2} of cell 1, and {3,4} of cell 2. We compute $G_{\mathscr{D}}(\cdot)$-terms using the convolution operator (see (1)):

$$G_{\mathscr{D}}(n_{\mathscr{D},8}, n_{\mathscr{D},9}) = \sum_{n_8=0}^{n_8=n_{\mathscr{D},8}} \sum_{n_9=0}^{n_9=n_{\mathscr{D},9}} G_1(n_8, n_9) \times G_2(n_{\mathscr{D},8} - n_8, n_{\mathscr{D},9} - n_9).$$

Note that part type 8 is fully covered by subset $\mathscr{D}$, while part 9 is still partially covered. Hence $g_{\mathscr{D}}(\cdot)$-terms need to be computed only for $n_{\mathscr{D},8} = N_8$, whereas for part 9 they need to be computed for its full population range. This needs $(N_8 + 1)(N_9 + 1)(N_9 + 2)/2$ operations, or, 9 operations and 2 storage locations. Part types 1, 2, 3 and 4 are fully covered by the children of node 4, and hence they do not explicitly enter computations at this node.

3. *Computation of* $G(N)$. At the root node we compute the value of the normalisation constant of the system using the convolution operator on the precomputed values of nodes 3 and 4. (Part 9 is the only part type partially covered at nodes 3 and 4, which becomes fully covered at node 5).

$$G(N) = G_5(N_9) = \sum_{n_9=0}^{N_9} G_3(n_9) \times G_4(N_9 - n_9).$$

This step needs 2 operations, and 1 unit of storage.

As an illustration of the performance measures obtained from the tree convolution algorithm, we present in Table 5 the utilisations of each of the machines by the 9 part types produced by the CMS, and in Table 6, the corresponding mean queue lengths.

Table 6
Tree CQN method: Mean queue lengths

Part type	Machine					
	1	2	3	4	5	6
1	9.101	0.899	–	–	–	–
2	7.571	0.429	–	–	–	–
3	–	–	8.345	1.655	–	–
4	–	–	9.497	0.503	–	–
5	–	–	–	–	1.612	3.388
6	–	–	–	–	1.612	3.388
7	–	–	–	–	1.936	3.064
8	1.037	0.0685	0.821	0.0739	–	–
9	0.1855	0.012	0.1462	0.0129	0.1334	0.5099

Table 7
Comparison of computation requirements: Sequential vs. tree convolution

	Evaluation of $G(N)$	
	Sequential convolution	Tree convolution
Space	15.52×10^6	1755
Operations	838.34×10^6	14027

If we use static storage allocation [16], retaining all the required storage throughout the computation of $G(N)$, we need a total of about 4500 storage locations and about 30000 operations only; these figures compare very favourably with the requirements of the unmodified sequential algorithm; see Table 7.

3.2. Tree mean-value analysis algorithm

The major difficulties encountered in implementing unmodified convolution algorithms, viz., the high possibility of numerical overflow or numerical underflow, and the complicated programming required to compute the performance measures as functions of the normalisation constant [21], are present in tree convolution also. Although initial scaling and dynamic scaling [15] may be used to alleviate the numerical difficulties, they do not remove the possibility of numerical overflows. The Mean-Value Analysis (MVA) algorithm [21] developed by Reiser and Lavenberg directly computes the performance measures in a closed product form CQN, avoids the numerical difficulties encountered in convolution, and is easy to implement. The space requirement of MVA is $M \times \prod_{j=1}^{j=K}(N_j + 1)$ storage units and the operations count is $M \times K \times \prod_{j=1}^{j=K}(N_j + 1)$, which are of the same order as those of convolution; hence MVA suffers from the difficulty of excessive computational requirements in the same way as convolution. The tree version of the MVA algorithm [12,23], analogous to tree convolution, can be used for performance evaluation of cellular FMSs with significant computational savings as in tree convolution.

Table 8
Comparison of computational requirements: Unmodified vs. tree MVA

	Evaluation of all performance measures	
	Sequential MVA	Tree convolution
Space	93.15×10^6	4500
Operations	1676.7×10^6	30000

The tree MVA algorithm uses the same overall scheme of computations as tree convolution. The concept of partial, full or noncoverage of a part type by a subset of machines is identical. It uses the same tree planting method and the tree traversal process. The array indices over which terms are computed and retained are identical. The main difference between the two is that tree MVA directly computes arrays of performance measures (like throughput of part type, or the waiting time of a part type), instead of the G-arrays of tree convolution; at an internal node, the MVA merge operator [12] is used to merge the arrays of the two child nodes, instead of the convolution operator; at leaf nodes of the binary tree (which indicate cells), the unmodified version of MVA is used to initialise the relevant arrays of performance measures. The application of tree MVA achieves computational reductions of the same magnitude as are achieved by applying the tree version to convolution. We outline below the tree MVA algorithm as applied to cellular FMSs. The derivation and physical interpretation of the underlying mathematics appear in [23] and [12].

1. *Tree planting.* Same as in tree convolution.

2. *Tree traversal.* We use the same post-order method of traversal of the binary tree.

- At a leaf node, corresponding to a cell c, we calculate the desired performance measures (throughput and queue length) for each part using the cell at each machine in cell c, over the range of indices $\mathbf{0} \leqslant \boldsymbol{n}_c \leqslant \boldsymbol{N}_c$. Terms are retained over the population range of the rare parts, with the corresponding index for local parts equal to their system population.
- At an internal node the performance measures of the child nodes are combined by the MVA merge operator to give the performance measures of the system formed by the union of the systems corresponding to the child nodes.

3. At the root node, we obtain the desired system performance measures by combining the results of the two children.

We outline below the application of tree MVA to the cellular FMS example described in the previous subsection.

- We use the same binary tree as in tree convolution, see Figure 2.
- The tree traversal is the same as in convolution, post-order traversal starting from the root node.

1. At node 1, representing cell 1, we compute the throughput of each of the parts 1, 2, 8 and 9 in the subnet of machines {1,2}, and the mean queue lengths of each of these parts at each of the machines in the cell, over the population range of the rare part types 8 and 9, corresponding to the full population of the local part types 1 and 2. Analogous computations are carried out for cells 2 and 3.

2. At an internal node, say node 3 representing the merging of cells 1 and 2 (forming the subnet of machines {1,2,3,4}), we apply the MVA merge operator. Since part 8 becomes fully covered by this subnet, we retain terms for the full population range of rare part 9, corresponding to the full population of part 8, for future computations.

- All system performance measures are obtained at the root node.

4. Decomposition of generalised stochastic Petri net models of CMSs

4.1. Generalised stochastic Petri nets

Petri nets are a formalism for the modelling and analysis of systems possessing concurrency, synchronisation, and asynchronous activities. We shall restrict ourselves to an informal description of Petri nets here; for a formal definition, extensions, and applications, see [19]. A Petri Net (PN) is a directed, bi-partite graph, consisting of two types of nodes, called *places* (drawn as circles) and *transitions* (drawn as bars). An *input arc* (respectively, *output arc*) is a directed arc from a place (transition) to a transition (place). Places represent conditions and transitions represent activities. *Tokens* (drawn as dots) in a place represent the truth value of a condition. A *marking* of a PN gives the number of tokens in each of its places. A transition is said to be *enabled to fire* in a given marking, if each place connected to this transition by an input arc contains at least one token. The *firing* of a transition removes a token from each place connected to it by an input arc and deposits one token in each place connected by an output arc, leading to a new marking. An initial marking (or, state) specifies the starting state of the PN; the *reachability set* of the PN markings is the set of all markings (states) that are obtainable by firing zero or more transitions starting from the initial marking.

Generalised Stochastic Petri Nets (GSPNs) are an important class of Petri nets, introduced by Ajmone Marsan et al. [1]. In a GSPN, a stochastic (exponential) duration is associated with the firing of certain transitions, called *timed* transitions (drawn as thick bars); the rest of the transitions fire instantaneously without any time delay, and are referred to as *immediate* transitions (drawn as thin bars). A timed transition represents a time-consuming activity, e.g., processing a workpiece or moving a workpiece, and an immediate transition represents nontime-consuming instantaneous decisions, e.g., deciding the next machine in the routing of a workpiece. A marking of a GSPN where no immediate transitions are enabled is called a *tangible* marking; a marking where at least one immediate transition is enabled is called a *vanishing* marking. It has been shown that a GSPN can be converted to and analysed as an underlying continuous-time Markov chain [1]. GSPNs have been succesfully used to model various kinds of systems like multiprocessor systems [1] and flexible manufacturing systems [25]. The reachability set of a GSPN must be finite if it is to be analysed as a CTMC.

4.2. GSPN model of cellular FMSs

The GSPN model of the cellular FMS is the combination of the models of each of the cells; the GSPN model of a cell is in turn a combination of the model of each of the machines in the cell. The transitions corresponding to part movement forms the link between models of the machines; the cells are connected by the transitions corresponding to the inter-cell movement of the rare parts.

The GSPN model of the machine i ($1 \leqslant i \leqslant M$) appears in Figure 3. A token in the place PMACHINE_i indicates that the machine is free. Tokens in the place $\text{PWAIT}_{i,j}$ where $j \in \mathcal{J}_i$ indi-

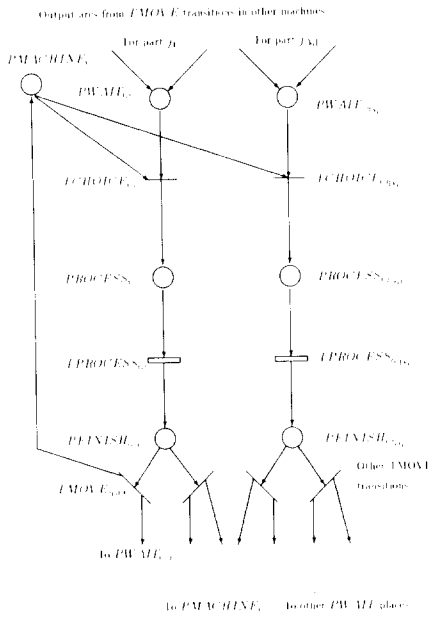

Figure 3. GSPN model of machine i

cate the number of parts of type j waiting to be processed on this machine. The machine cannot remain idle when workpieces are waiting. When the machine becomes free, it chooses a workpiece among the waiting part types with a probability proportional to the number of waiting workpieces of that type; this choice is represented by the immediate transitions $\text{TCHOICE}_{i,j}$. When enabled, the probability of firing of the transition $\text{TCHOICE}_{i,j}$ is given by $\text{Prob}[\text{TCHOICE}_{i,j}] = \#(\text{PWAIT}_{i,j})/\Sigma_{j_1 \in \mathcal{J}_i} \#(\text{PWAIT}_{i,j_1})$, where #(PLACE) gives the number of tokens in the place PLACE. The firing of the transition $\text{TCHOICE}_{i,j}$ deposits a token in the place $\text{PROCESS}_{i,j}$; at this stage only the timed transition $\text{TPROCESS}_{i,j}$ is enabled in the GSPN model of machine i. The firing time of the transition $\text{TPROCESS}_{i,j}$ is exponentially distributed with mean value $\tau_{i,j}$ corresponding to the mean processing time of a workpiece of type j on machine i. When the transition $\text{TPROCESS}_{i,j}$ fires, a token is deposited in the place $\text{PFINISH}_{i,j}$ indicating that the processing of a type-j part has been completed by machine i; the immediate transitions $\text{TMOVE}_{j;i,i_1}$ corresponding to moving the type-j workpiece from machine i to machine i_1 are enabled. The probability of firing of $\text{TMOVE}_{j;i,i_1}$ is p^j_{i,i_1}, the routing probability for a part-j transfer from machine i to machine i_1. The reachability set for the GSPN, the set of tangible states S_T, and the underlying CTMC are obtained by standard methods [1].

The probabilistic part selection policy followed here is easily seen to be the Service-in-Random-Order (SIRO) policy. Under the assumption that all the part types have identical, exponentially distributed processing times at a particular machine, the same product form structure [3] as described in the previous section (for a system with all machines operating under FCFS) holds for a much more general queueing discipline [13, Chapter 3; 17; 9, Section 2.1.4] FCFS and SIRO are special cases of this general discipline. The queueing model and GSPN model will be identical under this assumption, and will yield the same results (if the GSPN model is analysed exactly). It must be noted that in the GSPN model different part types *can* have processing times of different means at machine i, whereas in the queueing model, product form will be lost, and we have to use approximate analyses.

4.3. Analysis of the CTMC: State aggregation and decomposition

The state aggregation of the CTMC is based on the earlier observation that the frequency of intra-cell part movement is at least an order of

magnitude higher than the frequency of intercell part movement. An *aggregate state* of the CTMC is defined as one in which *the number of work-pieces of each rare part type present in each cell remains fixed.* (For a discussion of time scale aggregation of systems, see the fundamental work of Courtois [10].)

An aggregate state is uniquely characterised by a $C \times |\mathscr{R}|$-matrix $\boldsymbol{x}$, where the entry $x_{c,j}$ gives the number of rare parts of type j in cell c; $\boldsymbol{x}$ is referred to as a *configuration* of the distribution of the rare parts among the cells. (We can interpret $\boldsymbol{X}$ as a combination of $\boldsymbol{C}$ vectors, one for each cell, as $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_C)$, where the vector $\boldsymbol{x}_c$ gives the number each type of rare part present in cell $\boldsymbol{c}$ in the configuration $\boldsymbol{x}$). The set of all such configurations is given by

$$H = \left\{ \boldsymbol{x} \,|\, \forall j \in \mathscr{R}, \sum_{c=1}^{C} x_{c,j} = N_j; \ \forall j, \ x_{c,j} \geqslant 0 \right\} \quad (5)$$

We define the aggregate state $S_T(\boldsymbol{x})$ associated with the configuration $\boldsymbol{x} \in H$ as the set of those markings in S_T where the configuration defined by $\boldsymbol{x}$ holds:

$$S_T(\boldsymbol{x}) = \Big\{ s \,|\, s \in S_T; \ \forall c, \ \forall j \in \mathscr{R}_c, \sum_{i \in \mathscr{M}_c} \big[\#(\text{PROCESS}_{i,j}) + \#(\text{PWAIT}_{i,j}) \big] = x_{c,j} \Big\}. \quad (6)$$

At the fast time scale, corresponding to the time between intra-cell part moves, we ignore the inter-cell part moves at the lower time scale; this is equivalent to breaking down the GSPN model into individual GSPN models of the cells, by ignoring the transitions representing inter-cell part movement. The model of a cell is a much smaller system than a model of the entire cellular FMS. The following scheme is used to analyse the system (a similar framework for state aggregation and decomposition of queueing networks is discussed in [4]):

1. *Equivalence.* The CTMC is converted into an equivalent, aggregate CTMC whose state space S_{AGG} corresponds to the set of aggregate states:

$$S_{AGG} = \{ S_T(\boldsymbol{x}) \,|\, \boldsymbol{x} \in H \}.$$

The marginal probability of an aggregate state $S_T(\boldsymbol{x})$, $\boldsymbol{x} \in H$ is given by:

$$\text{Prob}[S(\boldsymbol{x})] = \sum_{s^x \in S_T(\boldsymbol{x})} \text{Prob}[s^x]. \quad (7)$$

2. *Decomposition.* Given that the system remains in the configuration $\boldsymbol{x} \in H$, there is no interaction between the cells due to the absence of intercell moves. Each cell behaves independently under this condition, and the system $S_T(\boldsymbol{x})$ can be analysed cell by cell. Specifically, an arbitrary state $s^x \in S_T(\boldsymbol{x})$ can be written in terms of its cell-by-cell components, as $s^x = [s_1^x, s_2^x, \ldots, s_C^x]$. Since these components are independent of each other, we can obtain $\text{Prob}[s^x \,|\, \boldsymbol{x}]$ from

$$\text{Prob}[s^x \,|\, \boldsymbol{x}] = \prod_{c=1}^{C} \text{Prob}[s_c^x \,|\, \boldsymbol{x}]. \quad (8)$$

Further, since the component s_c^x depends only on the portion $\boldsymbol{x}_c$, we have

$$\text{Prob}[s_c^x \,|\, \boldsymbol{x}] = \text{Prob}[s_c^x \,|\, \boldsymbol{x}_c]$$

(8) can be simplified as:

$$\text{Prob}[s^x \,|\, \boldsymbol{x}] = \prod_{c=1}^{C} \text{Prob}[s_c^x \,|\, \boldsymbol{x}_c]. \quad (9)$$

To maintain the recurrent nature [1] of the GSPN model of the isolated cells, the following technical detail needs attention: solving the GSPN models of the cells in isolation is equivalent to neglecting the immediate transitions corresponding to the intercell moves of the rare parts. In each cell (say, cell c), we add the probability of inter-cell move of a rare part (say, part j) from a machine (say, machine i) in that cell to the probability of that part being routed back to the same machine, i.e., $p_{i,i}^j$ is replaced by $p_{i,i}^j + \sum_{i_1 \notin \mathscr{M}_c} p_{i,i_1}^j$. Since we have neglected certain transitions, an error of the order of the maximum degree of coupling ([10, p. 13]) between the aggregate states is caused in the computations of the probabilities of the markings $s \in S_T$. The rate of the neglected transitions being orders of magnitude smaller than the rates of the transitions within the aggregate states, the error caused is negligble [10].

3. *Marginal probabilities of aggregate states.* The marginal probabilities are obtained by solving an aggregated CTMC whose state space is the set of aggregate states. The rates of transitions

between aggregate states are determined as follows: a transition occurs from aggregate $S_T(x)$ to aggregate $S_T(y)$, x, $y \in H$, $x \neq y$, if and only if configuration y is obtained from configuration x by the inter-cell movement of a rare part j from a cell c to another cell d, where j, c, d are unique and $c \neq d$. The rate $r_{AGG}(S_T(x), S_T(y))$ of this transition equals the sum of the rates of transitions corresponding to transferring a type-j part form any $i_1 \in (\mathcal{M}_c \cap \mathcal{X}_j)$ to any $i_2 \in (\mathcal{M}_d \cap \mathcal{X}_j)$. Let $S_{c,i_1,j} \subseteq S_T(x)$ denote the set of states in $S_T(x)$ where a type-j part is being processed on machine $i_1 \in \mathcal{M}_c$, i.e.,

$$S_{c,i_1,j} = \left\{ s \mid s \in S_T(x) \text{ and } \left(\#(\text{PROCESS}_{i_1,j}) = 1 \right) \right\}.$$

The rate $r_{AGG}(S_T(x), S_T(y))$ is given by:

$$r_{AGG}(S_T(x), S_T(y)) = \sum_{i_1 \in (\mathcal{M}_c \cap \mathcal{X}_j)} \sum_{i_2 \in (\mathcal{M}_d \cap \mathcal{X}_j)} \sum_{s \in S_{c,i_1,j}} \text{Prob}[s_c \mid x_c] \times \frac{1}{\tau_{i_1,j}} \times p^j_{i_1,i_2}. \quad (10)$$

4.4. Performance measures

A desired performance measure $g(i, j)$ for part j at a machine i is obtained from

$$g(i, j) = \sum_{s \in S_T} f(s) \quad (11)$$

where $f(\cdot)$ is an appropriately defined function on the state space of the underlying CTMC.

In an aggregate state $S_T(x)$, a performance measure $g(i, j \mid x)$ for part j at machine i, where $i \in \mathcal{M}_c$, is independent of the status of cells other than c. Thus

$$g(i, j \mid x) = g(i, j \mid x_c). \quad (12)$$

We can modify the function $f(\cdot)$ to $f(\cdot \mid x_c)$ to reflect the conditioning on the current aggregate state. The conditioning of the performance measure on the aggregate state is removed using the relation

$$g(i, j) = \sum_{x \in H} g(i, j \mid x) \times \text{Prob}[S_T(x)]. \quad (13)$$

Since the performance measures $g(i, j \mid x)$ depends only on the number of rare parts in cell c_1 (from (12)),

- it is sufficient to solve the GSPN model of each cell c for each different configuration of rare parts x_c pertaining to this cell only;
- we can further simplify (13) to:

$$g(i, j) = \sum_{\{x_c \mid x \in H\}} g(i, j \mid x_c) \times \text{Prob}[S_T(x_c)]. \quad (14)$$

Here, $\text{Prob}[S_T(x_c)]$ is the sum of the marginal probabilities for those aggregate states $S_T(x)$ where the rare part configuration in cell c is x_c, and it is evaluated by

$$\text{Prob}[S_T(x_c)] = \sum_{\{y \in H \mid y_c = x_c\}} \text{Prob}[S_T(y)]. \quad (15)$$

The function $f(\cdot)$ for obtaining the performance measures of interest, viz., utilisation $U_{i,j}$, throughput $T_{i,j}$, mean queue length $Q_{i,j}$, and the mean waiting time $W_{i,j}$, for part j at machine i located in cell c, are defined as follows ([25], Sect 2.1):

Utilisation.

$$f(s_c \mid x_c) = \begin{cases} \text{Prob}[s_c \mid x_c] & \text{if } \#(\text{PROCESS}_{i,j}) = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

Queue length.

$$f(s_c \mid x_c) = \text{Prob}[s_c \mid x_c] \times \left[\#(\text{PWAIT}_{i,j}) + \#(\text{PROCESS}_{i,j}) \right]. \quad (17)$$

Throughput.

$$T_{i,j} = \frac{U_{i,j}}{\tau_{i,j}}. \quad (18)$$

Waiting time.

$$W_{i,j} = \frac{Q_{i,j}}{T_{i,j}}. \quad (19)$$

From (13), we find that it is *not* necessary to obtain the probabilities of the system state vectors $s \in S_T$ for evaluating the performance mea-

sures, or to build the set of entire set of system states. It is sufficient to obtain the much smaller set of tangible markings and the transition rate matrix of the GSPN subnets of individual cells, and solve for their steady-state probabilities.

4.5. Algorithm for the decomposition analysis of the GSPN model of a CMS

We outline below the methodology for the analysis:

- Identify the set H giving the various configurations of rare part distribution among the cells.
- *for each* $x \in H$ *do*
  - *for each* cell c *do*
  - *begin*
  - *if* (cell c has not been solved for configuration x_c) *then*

1. obtain the probabilities Prob$[s_c \mid x_c]$ of the states in the GSPN model of cell c under configuration x_c;
2. calculate the values of the performance measures $U_{i,j}$, $Q_{i,j}$ using the values of Prob$[s_c \mid x_c]$ and (16) and (17), for the configuration x_c;
3. obtain the rate of transition $r_{\mathrm{AGG}}(S_{\mathrm{T}}(x), S_{\mathrm{T}}(y))$from configuration x to another configuration $y \in H$, if y can be reached from x by the unique inter-cell movement of a rare-type workpiece (see (10)).

- *end*
- Solve the system of aggregate states, using the transition rates between the aggregate states obtained in the previous step, and obtain the marginal probabilities.
- Uncondition the dependence of the performance measures on the configurations using (14) and (15)
- Calculate the values of the throughputs and waiting times from (18) and (19).

4.6. Complexity of the decomposition algorithm

We concentrate on finding the computational requirements to determine the utilisations and queue lengths, since the throughputs and waiting times are obtained easily (from (18) and (19)), with little additional computation. Since an arbitrary part j visits $|\mathscr{R}_j|$ machines, we have a total of $\Omega = 2 \times \sum_{j=1}^{K} |\mathscr{R}_j|$ different performance measures to compute.

4.6.1. Complexity of the exact analysis

The number of states in the GSPN model of the system (*without decomposition*) is given by

$$|S_{\mathrm{T}}| = \sum_{x \in H} |S_{\mathrm{T}}(x)|, \tag{20}$$

since we have to consider all possible interactions due to part moves. The cardinality of the aggregate state $S_{\mathrm{T}}(x)$ equals the product of the the number of markings in the GSPN subnets of each of the cells in the configuration $x \in H$. The cardinality of the GSPN model of the system, given by (20), can be very large even for moderately sized problems like our example cellular FMS (see Section 4.7). Since the set of tangible markings forms a CTMC, the steady-state probability for each state in S_{T} can be obtained by solving a set of $|S_{\mathrm{T}}|$ linear equations, which requires $O(|S_{\mathrm{T}}|^3)$ time and $O(|S_{\mathrm{T}}|^2)$ space. Each performance measure requires one operation (evaluating the function $f(\cdot)$ and applying (11)) on each tangible marking in S_{T}. Hence the total time requirement is $O(|S_{\mathrm{T}}|^3 + (\Omega \times |S_{\mathrm{T}}|))$. Since usually $\Omega \ll |S_{\mathrm{T}}|$, the time requirement is $O(|S_{\mathrm{T}}|^3)$.

4.6.2. Complexity of the decomposition analysis

The number of different performance measures, Ω_c, associated with cell c is given by

$$\Omega_c = 2 \times \sum_{j \in (\mathscr{L}_c \cup \mathscr{R}_c)} |\mathscr{M}_c \cap \mathscr{R}_j|.$$

In the analysis by decomposition, we need to compute and retain these performance measures for each different configuration of rare parts in each cell c, for use in the aggregation step (cf. (14)). The space requirement for storing all these performance measures is $\sum_{c=1}^{C} \Delta_c \times \Omega_c$, where Δ_c is the cardinality of the set of different configurations for cell c.

Although each cell has to be solved individually in the given configuration of rare parts, the steady-state probabilities computed for the cell in the current rare part configuration are no longer needed once the performance measures are computed. Hence the space required to solve for these steady-state probabilities may be released for future computations. The maximum state

space cardinality of any individual cell is

$$|S_{\mathrm{DECOMP}}| = \mathrm{MAX}_{1 \leq c \leq C} |S_{\mathrm{DECOMP},c}| = \mathrm{MAX}_{1 \leq c \leq C} [\mathrm{MAX}_{x \in H} |\{s_c \mid s \in S(x)\}|]$$

where $|S_{\mathrm{DECOMP},c}|$ is the maximum state space requirement of a particular cell. Since usually $S_{\mathrm{DECOMP}} \gg \Sigma_{c=1}^{C} \Delta_c \times \Omega_c$, and the underlying CTMC for the GSPN model of a cell can be solved as a linear system, the space requirement is $O(|S_{\mathrm{DECOMP},c}|^2)$. In practice, the transition matrix of the underlying CTMC of the GSPN model of a cell is sparse; using sparse matrix representation by dynamic data structures, the storage space requirement is far smaller than $|S_{\mathrm{DECOMP},c}|^2$.

Analogous to the case of exact analysis, the time requirement to solve a cell c (over all iterations) is $O(\Delta_c \times |S_{\mathrm{DECOMP},c}|^3)$, and the total time requirement of the decomposition algorithm is $\mathcal{T}_{\mathrm{DECOMP}} = O(\Sigma_{c=1}^{C} \Delta_c \times |S_{\mathrm{DECOMP},c}|^3)$.

4.7. Example

We outline below the GSPN analysis of the example cellular FMS (described in Section 2) by the methodology presented in Section 4.5.

The set of aggregate states for this example is given by the various configurations of the rare part types 8 and 9 in the three cells. The aggregate state can be represented by a 2-by-3 matrix, where the first row corresponds to rare part 8 and the second row corresponds to rare part 9, and columns 1, 2, 3 correspond to cells 1, 2, 3, respectively. An entry of the matrix gives the number of rare parts of the corresponding rare part in the relevant cell, e.g., the entry in the (1,1) position gives the number of rare parts of type 8 in cell 1. The set of rare part configurations is:

$$x_1 = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

$$x_3 = \begin{pmatrix} 0 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad x_4 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

$$x_5 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad x_6 = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

$$x_7 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad x_8 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$x_9 = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

It is sufficient to analyse the GSPN model cell for each different configuration of rare parts found in it; we need not analyse it for every $x \in H$. As an illustration, for cell 3, the configurations x_1 through x_6 are equivalent, since all of these correspond to a situation where there are no rare-type parts being processed in cell 3; it suffices to evalute the performance of cell 3 once for these six configurations. Such simplifications are possible since a rare part (in our example, rare part 8) may not visit all the cells in the system, a common situation in cellular FMSs.

From the solution of the GSPN subnet of each cell, we determine:

- The transition rates between the set of aggregate states $S_{\mathrm{AGG}}(x)$, using (10). For example, the transition rate from the aggregate state $S_{\mathrm{AGG}}(x_1)$ to the aggregate state $S_{\mathrm{AGG}}(x_2)$ corresponds to the rate of inter-cell movement of a workpiece of rare part type 8 from cell 1 to cell 2. Other transition rates are obtained analogously.
- The performance measures of interest for each cell in the given configuration.

The solution of the aggregated CTMC gives the marginal probabilities of the 9 aggregate states. When performance measures of interest are calculated using (14) and (16)–(19), we can once again take advantage of the equivalence of certain configurations for a cell. For example, the utilisation of part 5 at machine 6 of cell 3 can be computed using (14) as

$$\mathrm{UTIL}_{6,5} = \sum_{i=1}^{i=9} \mathrm{Util}(6, 5 \mid x_i) \times \mathrm{Prob}(S_{\mathrm{AGG}}(x_i)).$$

Since configurations x_1 through x_6 are all equivalent to each other for cell 3, and so are configurations x_7 through x_9, the above computation may be simplified to:

$$\mathrm{UTIL}_{6,5} = \left(\sum_{i=1}^{i=6} \mathrm{Prob}(S_{\mathrm{AGG}}(x_i)) \right) \times \mathrm{Util}(6, 5 \mid x_i) + \left(\sum_{i=7}^{i=9} \mathrm{Prob}(S_{\mathrm{AGG}}(x_i)) \right) \times \mathrm{Util}(6, 5 \mid x_7).$$

Tables 9 and 10 summarise respectively the utilisations of and mean queue length at the various machines, for each part type, obtained by the decomposition analysis of the GSPN model.

Table 9
GSPN decomposition analysis: Utilisation of the machines

Part type	Machine					
	1	2	3	4	5	6
1	0.5059	0.3794	–	–	–	–
2	0.4260	0.1743	–	–	–	–
3	–	–	0.4367	0.5435	–	–
4	–	–	0.5148	0.1478	–	–
5	–	–	–	–	0.2591	0.3239
6	–	–	–	–	0.2591	0.3239
7	–	–	–	–	0.3364	0.2803
8	0.0574	0.02873	0.0431	0.0229	–	–
9	0.0106	0.005307	0.005307	0.005611	0.0212	0.0531

Since this example is exactly solvable as a PF-CQN model (Section 3.1.4), we obtain the error in the GSPN analysis by comparison with the results of the queueing analysis. Owing to space restrictions, we report the error figures only for the mean queue lengths.

Table 11 summarises the number of states of the GSPN model of each cell, for each configuration, for both the decomposition analysis and the exact analysis. The number of states of the analysis not using decomposition is the product of the number of states for each cell, as indicated by (20). We see that the maximum state space to be handled by the decomposition analysis is only 6202, far lesser than the nearly 30848 million states required to be handled in the exact analysis.

5. Conclusion

The main contribution of the paper is the development of an efficient decomposition-based analytical performance evaluation methodology for cellular FMSs using product form CQN models and GSPN models. Without recourse to de-

Table 10
GSPN decomposition analysis: Mean queue lengths at each of the machines. The magnitude of error is obtained by comparison with the PF-CQN result

Part type	Machine					
	1	2	3	4	5	6
1	9.109	0.891	–	–	–	–
|error|	0.0076	0.0076	–	–	–	
2	7.561	0.439	–	–	–	–
|error|	0.0103	0.0103	–	–	–	
3	–	–	8.360	1.64	–	–
|error|	–	–	0.0147	0.0147	–	–
4	–	–	9.463	0.537	–	–
|error|	–	–	0.0341	0.0341	–	–
5	–	–	–	–	1.606	3.394
|error|	–	–	–	–	0.598	0.0598
6	–	–	–	–	1.606	3.394
|error|	–	–	–	–	0.0598	0.0598
7	–	–	–	–	1.907	3.093
|error|	–	–	–	–	0.0287	0.0287
8	1.038	0.0677	0.820	0.0742	–	–
|error|	0.0013	0.00086	0.00074	0.0026	–	–
9	0.194	0.0124	0.1037	0.0179	0.1411	0.5308
|error|	0.0085	0.00044	0.0425	0.00503	0.0077	0.0208

Table 11
Comparison of the dimensions of the state space of the GSPN model: Decomposition approach versus exact analysis

Configuration	Number of states			
	Decomposition approach			Exact analysis
	cell 1	cell 2	cell 3	
x_1	5002	402	1338	2690.46×10^6
x_2	2902	1242	1338	4822.52×10^6
x_3	998	2201	1338	2939.05×10^6
x_4	1771	1242	1338	2943.04×10^6
x_5	998	3602	1338	4809.84×10^6
x_6	322	6202	1338	2672.04×10^6
x_7	1771	402	3752	2671.21×10^6
x_8	998	1242	3752	4650.66×10^6
x_9	322	2201	3752	2659.12×10^6
Max. no. of states		6202		30847.94×10^6

composition, analytical performance evaluation may not be feasible at all, as was illustrated by the example considered. The key merit of the product form analysis is that it is exact; the GSPN-based analysis can elegantly account for nonproduct form features and produces very good approximations. In the GSPN decomposition method, we also presented a systematic procedure for combining the performance measures of individual cells into system-wide performance measures. In both methods, the largest system to be handled at a time corresponds to a cell only, which has a state space many orders of magnitude smaller than that of the entire system. The methodologies presented here may be refined and extended in a number of ways. Possible refinements (which are not discussed in this paper due to size limitations) include the following:

- The use of *tree versions* of newer computational algorithms like RECAL [18] whose computational requirements are polynomial in the given number of part types, for a given number of machines, will be very useful for the manufacturing context, where we usually have more part types than machines.
- An efficient comparative evaluation of alternative cellular FMS configurations resulting from the application of different part-family-grouping/machine-cell-identification algorithms can be carried out by the methods developed in this paper.
- The methodology presented here can be adapted to the analysis of cellular FMS layouts, by simply modelling the material handling equipment of a cell as a 'machine' with suitable operating characteristics.
- The state space decomposition and aggregation performed in the GSPN analysis at the Markov chain level can be rendered more accurate by adapting more sophisticated aggregation/disaggregation methods (for example, see [24]). It is also possible to carry out the aggregation at the GSPN level [2].
- The individual cells can be modelled to a greater level of detail, e.g., by considering limited inventory storage area (i.e., limited number of buffers in each queue, leading to a model with blocking), equipment (machine, or MHS) failure and repair, etc. The overall decompostion–aggregation method of computations can follow the scheme presented here; only the solution methodology for individual cells have to account for the additional features. The accuracy of the decomposition approach, when the solution of the cells is itself approximate, is an issue to be investigated.

References

[1] Ajmone Marsan, M., Conte, G., and Balbo, G., "A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems", *ACM Transactions on Computer Systems* 2/2 (1984) 93–122.

[2] Ammar, H.H., and Rezaul Islam, S.M., "Time scale decomposition of a class of generalized stochastic Petri net models", *IEEE Transactions on Computers* 15/6 (1989) 809–820.

[3] Baskett, F., Mani Chandy, K., Muntz, R.R., and Palacios, F.G., "Open, closed and mixed networks of queues with different classes of customers", *Journal of the ACM* 22/2 (1975) 248–260.

[4] Brandwajn, A., "Equivalence and decomposition in queueing systems – A unified approach", *Performance Evaluation* 5 (1985) 175–186.

[5] Burbridge, J.L., *Group Technology and the Engineering Industry*, Mechanical Engineering Publications Limited, London, 1979.

[6] Buzen, J.P., "Computational algorithms for closed queueing networks with exponential servers", *Communications of the ACM* 16/9 (1973) 527–531.

[7] Buzacott, J.A., and Yao, D.D., "Flexible manufacturing systems: A review of analytical models", *Management Science* 32/7 (1986) 890–905.

[8] Conway, A.E., and Georganas, N.D., "RECAL – A new efficient algorithm for the exact analysis of multiple-chain closed queueing networks", *Journal of the ACM* 33/4 (1986) 768–791.

[9] Conway, A.E., and Georganas, N.D., *Queueing Networks – Exact Computational Algorithms,* Computer Systems Series, MIT Press, Cambridge, MA, 1989.

[10] Courtois, P.J., *Decomposability: Queueing and Computer Science Applications,* Academic Press, New York, 1977.

[11] Diesslin, R.L., "Group technology", in: D. Tijulenis and K.E. McKee (eds.), *Manufacturing High Technology Handbook,* Marcell Dekker, New York, 55–82.

[12] Hoyme, K.P., Bruell, S.C., Afshari, P.V., and Kain, R.Y., "A tree-structured mean value analysis algorithm", *ACM Transactions on Computer Systems* 4/2 (1986) 178–185.

[13] Kelly, F.P., *Reversibility and Stochastic Networks,* John Wiley, Chichester (UK), 1979.

[14] Kusiak, A., and Chow, W.S., "Decomposition of manufacturing systems," *IEEE Journal of Robotics and Automation* 4/5 (1988) 457–471.

[15] Lam, S.S., "Dynamic scaling and growth behaviour of queueing network normalisation constants", *Journal of the ACM* 29/2 (1982) 492–513.

[16] Lam, S.S. and Luke Lien, Y., "A tree convolution algorithm for the solution of queueing networks", *Communications of the ACM* 26/3 (1983) 203–215.

[17] Mani Chandy, K., and Martin, A.J., "A characterisation of product-form queueing networks", *Journal of the ACM* 30/2 (1983) 286–299.

[18] McKenna, J., "A new proof and a tree algorithm for RECAL", in: P.-J. Courtois and G. Latouche (Eds.), *The 12th IFIP International Symposium on Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE '87),* Elsevier Science Publishers, Amsterdam, 1988, 3–16.

[19] Murata, T., "Petri nets: Properties, analysis and applications", *Proceedings of the IEEE* 77/4 (1989) 541–580.

[20] Reiser, M., and Kobayashi, H., "Queueing networks with multiple closed chains: Theory and computational algorithms", *IBM Journal of Research and Development* 19/3 (1975) 283–294.

[21] Reiser, M., and Lavenberg, S.S., "Mean-value analysis of closed multichain queueing networks", *Journal of the ACM* 27/2 (1980) 313–322.

[22] Suri, R., "An overview of evaluative models for flexible manufacturing systems", *Annals of Operations Research* 3 (1985) 13–21.

[23] Tucci, S., and Sauer, C.H., "The tree MVA algorithm", *Performance Evaluation* 5 (1985) 187–196.

[24] Vantilborgh, H., "Aggregation with an error of $O(\epsilon^2)$", *Journal of the ACM* 32/1 (1985) 162–190.

[25] Viswanadham, N., and Narahari, Y., "Stochastic Petri net models for performance evaluation of automated manufacturing systems", *Information and Decision Technologies* 14 (1988) 125–142.

[26] Wemmerlov, U., and Hyer, N.L., "Cellular manufacturing in the US: A survey of users", *International Journal of Production Research* 27/9 (1989) 1511–1530.