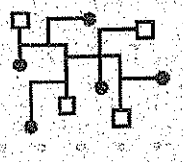


ISE

Industrial and Systems Engineering

A Multi-Level Multi-Layer Petri Net Approach to Modeling and Optimization of Production Systems

Gonzalo Mejia
and
Nicholas G. Odrey
Lehigh University

Report No. 02W-002

LEHIGH
University

200 West Packer Avenue
Bethlehem, PA 18015

A Multi-Level Multi-Layer Petri Net Approach to Modeling and Optimization of Production Systems

Gonzalo Mejía
Department of Industrial and Systems Engineering
Lehigh University
Bethlehem, PA 18105

Nicholas G. Odrey
Department of Industrial and Systems Engineering
Lehigh University
Bethlehem, PA 18105

Abstract

This paper presents a Petri Net-based multi-level multi-layer structure to model hierarchical manufacturing systems consistent with manufacturing control requirements. We obtain first a sequence of coordinated activities related to a comprehensive task decomposition. Tasks are then assigned to hierarchical levels depending on the complexity and planning horizon. In this paper we model the activities of the so termed workstation level using different classes of Petri Nets. We propose a formal mathematical description to construct the net and its state equations. Such equations and heuristic search are used to optimize the execution of the process plan.

Keywords. Color Petri Nets, Flexible Manufacturing Systems, Heuristic Search. Hierarchical Architectures.

1. Introduction

Many researchers advocate the use of Petri Nets as a mathematical tool for modeling FMS due to their capabilities to represent features such as asynchronous event-driven behavior, concurrency, non-determinism and resource sharing. In addition their graphical representation makes them attractive for designers and practitioners. As pointed by [8] one of the purposed of modeling is to achieve a comprehensive task decomposition that converts into a sequence of coordinated activities which in turn is translated into manufacturing commands for controllers at each hierarchical level of the FMS.

In our approach task decomposition is accomplished by assigning production activities to distinct levels of a hierarchical structure according to its complexity and planning horizon. In particular we follow the standards set by NIST [1] to assign tasks to hierarchical levels. In order to accommodate the complex nature of the manufacturing activities, a hybrid Petri Net approach is proposed to facilitate hierarchical task decomposition. In this paper we present a methodology to synthesize the classes of Petri Nets used in our approach. Our synthesis methodology guarantees structural properties and provides enough freedom to model a variety of manufacturing processes. Our approach includes the derivation of state equations used to track the evolution of the net in terms of the marking and the remaining processing time vector.

Another major issue is optimization of the operation of the manufacturing system at the workstation level. The primary activity of a workstation controller is to realize effectively the process plan. An adequate model of such process plan is a requirement for optimization. Petri Nets have not received much attention for solving optimization problems due in part to its state space explosion. Exhaustive search of the reachability graph is known to be a problem of exponential complexity. To overcome this problem recent approaches attempt to use Artificial Intelligence (AI) techniques [3] [5] [7] to the search the state space of Timed Petri Nets. In this paper we present a new heuristic that shows promising results.

2. Modeling and Synthesis of Manufacturing Activities using Petri Nets.

In this paper we propose a multi-layer Petri Net framework that uses Ordinary Petri Nets (OPN), Timed Petri Nets (TPN) and Timed Colored Petri Nets (TCPN) to model the activities at workstation level. The advantages of TCPN's are fully utilized at the highest abstract layer where similar features are best described by a coloring

scheme[8]. Lower layers are best represented by OPN's where explicit details need to be represented. Intermediate layers use TPN's for time critical analysis [8]. In this paper limit our model to a two-layer Petri Net. Typical activities at the workstation level include processing of parts, loading to and unloading from machines, and tool changing.

The first layer is modeled with a TCPN which has the purpose of modeling the following flexibility features:

- Routing: A part can be processed by the different machines of the workstation.
- Transportation: A part can be transferred by different material handling devices between machines.
- Sequencing: Machines can process parts in different orders.

Colors at the TCPN layer are defined by a part number (*p#*) and resource numbers (*rs#*). In addition a resource number is decomposed into several colors that represent specific resources. For example the set *rs* can include machine (*m#*), robot (*r#*) material handling (*h#*) and buffer colors (*b#*). Example: token carrying the color {*p1, m1, r1*} represents a part *pt1*, handled by robot *r1* and being machined by machine m1. In our scheme we use the traditional color "merging" described by [4]. Colors from input places are merged in transitions and assigned to output places according to pre-established coloring rules. Flexibility features require the definition of which color sets are allowed in the net. Processing or time constraints may forbid certain color sets from the net. For example merging colors from sets {pt1, pt2} and {m1, m2} will result in the color set {pt1m1, p2m2, p2m1, p2m2} but a processing constraint may eliminate the color p2m2. The second layer is modeled with TPN that results from unfolding the TCPN layer. At this layer a specific part sequence, resource allocation and timing for each operation are incorporated in the model. A key issue concerns as to how the layers should be constructed. In the following sections we present formal definitions that define the construction rules for such nets.

2.1 Petri Nets Definitions:

Timed Colored Petri Net (TCPN): A TCPN is 7-tuple G = (P, T, I, O, M_0, τ, C). Where P = set of places, T = set of transitions I = arcs (P × T) O = arcs (T × P), M_0 = Initial marking. P ∩ T = {0}.τ = Set of time delays associated with places. C = set of colors.

Node: A node is either a place p∈ P or a transition t∈ T

Start and End place sets:

$P_s = \{p\} / p \in P_s$ if $\| \bullet p \| = 0$ Start place

$P_e = \{p\} / p \in P_e$ if $\| p \bullet \| = 0$ End place

Arcs: I (p, t) = 1 implies that there is an input arc between p and t. O (t, p) = 1implies that there is an input arc between t and p. [11]

Elementary Path (EP): A sequence of nodes $x_1, x_2, \ldots x_n$, $n \geq 1$ such that ∃ arc (x_i, x_{i+1}) i = 1… n-1. An elementary path EP is denoted as EP (x_1, xn) [11]. An EP must contain at least two nodes. Elementary paths are also treated as sets. Therefore $x \in$ EP (x_1, x_n) means that x is a node of the elementary path EP (x_1, x_n). [11]

2.1.1 Definition: Sequence of Activities (SA)

A SA is a Petri Subnet that is intended to model the activities at a hierarchical level required to process a part or a job (the distinction depends on the hierarchical level) and it accounts for precedence relationships and alternative routings. Each sequence starts in one unique place and ends in another unique place. Places that belong to a SA are denoted as operational places. An EP containing only operational places is denoted as an Operational Elementary Path (OEP). Places representing resources or other ways of coordination are not accounted for. Example is shown in figure 1.

Definition: SA = G (P_{SA}, T_{SA}, I_{SA}, O_{SA}, M_{0SA},) which satisfies the following constraints:

- SA is a connected state machine. A state machine is a Petri Net G such that $\forall t \in T_{SA} / \| t\bullet \| = \| \bullet t \| = 1$
- $\| P_s \cap P_{SA} \| = 1$There is only one start place.
- $\| P_e \cap P_{SA} \| = 1$ There is only one end place.
- $\forall p \in P_{SA} \wedge \tau(p) \geq 0$, $\| \bullet p \| = \| p \bullet \| = 1$. There is only one incoming arc and one outgoing arc to any operational place whose associated processing time is greater then zero.
- $\forall p, p' \in T_{SA} / \neg\exists$ (EP(p, p') ∧ EP(p', p)) if $p \neq p'$.
- $\forall t, t' \in T_{SA} / \neg\exists$ (EP(t, t') ∧ EP(t', t)) if $t \neq t'$. These last two conditions ensure that there are no cycles.

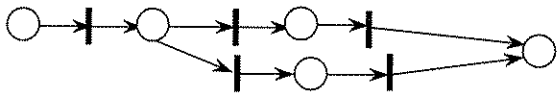


Figure 1 A valid SA.

Definition: X is a p-invariant if $X^T C = 0$ and $X > 0$. C is the incidence matrix of G.
Proposition: For any SA = G (P_{SA}, T_{SA}, I_{SA}, O_{SA}, M_{0SA}) there is only one p-invariant whose support is P_{SA}.
Proof: A SA is a state machine and therefore the number of tokens is constant. A subset $P' \subset P_{SA}$ is not a p-invariant support since $\exists$ t$\in$ T_{SA} that adds/removes a token to/from the set P'. A definition of SA is similar to that of Simple Sequential Process [2]. However there are two major differences. SA's contain timed places and SA's are not strongly connected state machines.

2.1.2 Definition: Set of Sequences of Activities (SSA).
SSA is an extension of SA. SSA is the Petri Net formed by several SA's. We assume there are a total of *N* SA's forming the SSA.
SSA is a G (P_{SSA}, T_{SSA}, I_{SSA}, O_{SSA}, M_{0SSA})
$P_{SSA} = \cup P_{SAi}$ / i=1…N. P_{SAi} is the set of places of SA_i.
$T_{SSA} = \cup T_{SAi}$ / i=1…N T_{SAi} is the set of transitions of SA_i.
$I_{SSA} = \cup I_{SAi}$ / i=1…N I_{SAi} is the set of input arcs of SA_i.
$O_{SSA} = \cup O_{SAi}$ / i=1…N O_{SAi} is the set of output arcs of SA_i.

2.1.3 Definition: Set of Activities with Resources (SSAR)
A SSAR represents the addition of resources to a set of sequential processes. The addition of resources implies addition of places, and arcs but not additional transitions. See figure 2 for an example.
Formal definition: SSAR = G ($P_{SSA} \cup R$, T_{SSA}, $I_{SSA} \cup I_R$, $O_{SSA} \cup O_R$, $M_{0SSA} \cup M_{0R}$)
R= {*r*} is the set of resource places. I_R = Set of input arcs R × T_{SSA}. O_R = Set of output arcs T_{SSA} × R. M_{0R} Initial marking of r$\in$ R.

Allocation of resources: Each resource place *r* is associated with a set of pairs of transitions $\{(t^r_{a1}, t^r_{b1}), (t^r_{a2}, t^r_{b2}), \ldots (t^r_{anr}, t^r_{bnr})\}$. A resource *r* is allocated to a sequence of activities when t^r_{ai} fires and is released when t^r_{bi} fires.
Formal definition: $\forall r \in R \; \exists \; \{(t^r_{ai}, t^r_{bi})\}$: / $t^r_{ai}, t^r_{bi} \in T_{SSA}$. i=1…n_r. [11]
n_r is the number of pairs (t^r_{ai}, t^r_{bi}). $T^r_a = \{t^r_{ai}\}$ and $T^r_a = \{t^r_{bi}\}$, i=1…n_r.
A SSAR is a Petri Net G in which the following conditions are satisfied:
1. $t^r_{ai} \neq t^r_{bi}$ i=1…n_r , $\forall r \in R$
2. $I(r, t^r_{ai}) = O(r, t^r_{bi}) = 1$. There exists a directed arc between the resource place and the transition t^r_{ai} and between t^r_{bi} and the resource place. $\forall r \in R$
3. $O(r, t^r_{ai}) = I(r, t^r_{ai}) = 0$. There does not exist a directed arc between the resource place *r* and a transition t^r_{bi} or between t^r_{ai} and a resource place *r*.
4. $\exists$ EP (t^r_{ai}, t^r_{bi}) / if $p \in$ EP (t^r_{ai}, t^r_{bi}) then $p \in P_{SSA}$. There exists a path constituted by only operational places between transitions t^r_{ai} and t^r_{bi}. EP (t^r_{ai}, t^r_{bi}) is an OEP. $\forall r \in R$.
5. If $\exists$ an OEP (t^r_{ai}, t^r_{aj}) then $t^r_{bi} \in$ OEP(t^r_{ai}, t^r_{aj}) i, j $\in \{1..n_r\}$. A resource cannot be re-allocated before being released. Conversely: If $\exists$ an OEP (t^r_{bi}, t^r_{bj}) then $t^r_{aj} \in$ OEP(t^r_{bi}, t^r_{bj}) i, j $\in \{1..n_r\}$. A resource cannot be released twice before being re-allocated. $\forall r \in R$
6. $\forall$ OEP (t^r_{au}, p) / $t^r_{bu} \notin$ OEP (t^r_{au}, p), $\exists$ OEP(p, t^r_{bu}). $t^r_{au}, t^r_{bu} \in T_{SSA}$, $p \in P_{SSA}$ u$\in \{1 \ldots n_r\} \forall r \in R$
Conversely $\forall$ OEP (p, t^r_{bu}) / $t^r_{au} \notin$ OEP(p, t^r_{bu}) , $\exists$ OEP(t^r_{au}, p). $t^r_{au}, t^r_{bu} \in T_{SSA}$, $p \in P_{SSA}$, u$\in \{1..n_r\}$ These conditions guarantee that if a sequence of activities has alternative processes (branches), resources are allocated and released on each branch.

2.1.4 Structural Properties
It can be proved with the theory of the p-invariants that a SSAR is bounded. However liveness cannot be guaranteed unless a very strict allocation policy or a well-defined pre-established control logic are adopted (e.g [10]). Since we

do not want to impose too many restrictions to the net modeling we are investigating techniques to maneuver out of deadlock states.

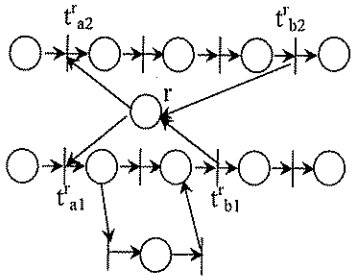


Figure 2 a valid SSAR.

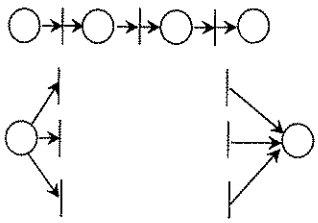
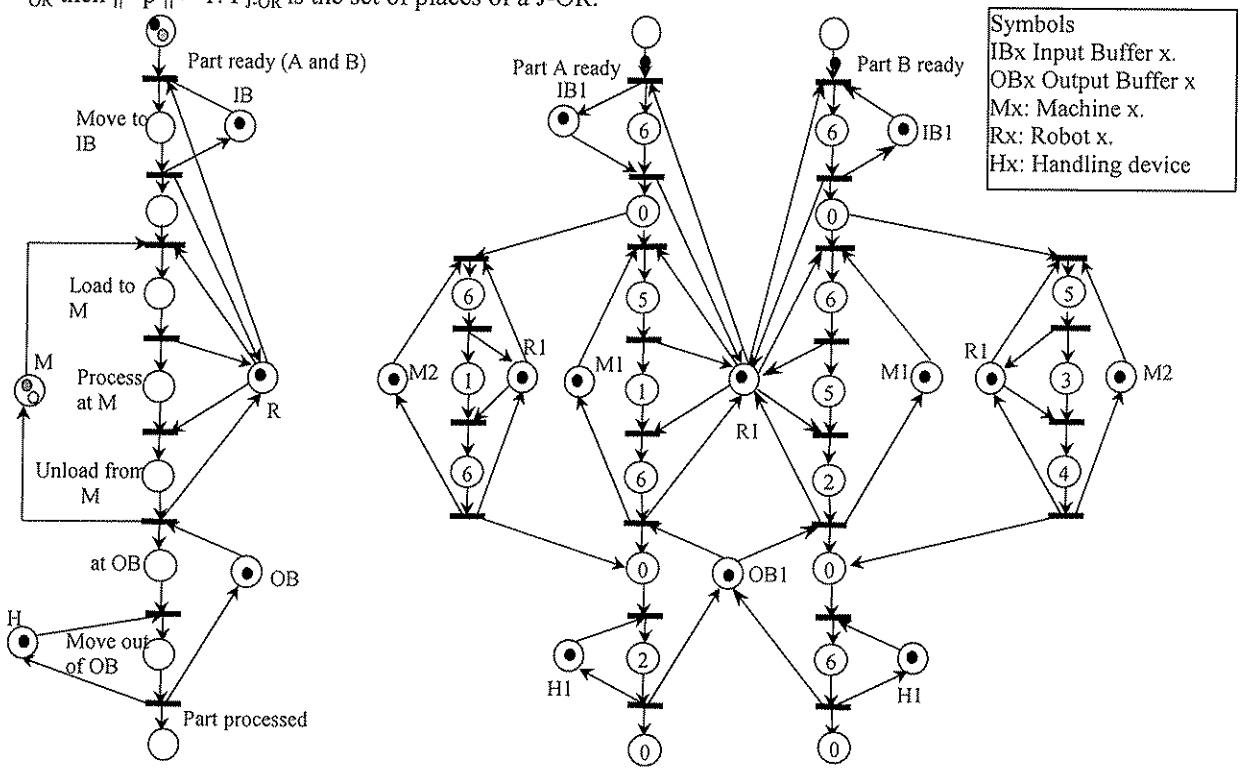


Figure 3. (a) Linear sequence (b) a Split-OR (c) Join-OR

2.2 Petri Net Synthesis

The concepts presented in this section apply to OPN's but can be readily extended to TPN's and TCPN's. To synthesize a Petri Net we consider a hybrid methodology between top down and bottom up approaches: Top down in this context refers to the subsequent refinement of places and transitions into more detailed activities at different levels and layers. Bottom up synthesis consists of having modular constructs that are subsequently connected to form a coherent structure. We adopt the concept of modules in conjunction with the concepts of SSAR. The modules (i) have neither resources places nor arcs to resource places and (ii) do not represent specific operations. Instead we use three generic modules that can accommodate a variety of situations. These modules are adapted from [9] and [11]. See figure 3.

- **Linear Sequence (LS):** A LS is a connected state machine that satisfies the conditions $\forall\ p \in P_{LS}\ \|\bullet p\| \leq 1 \wedge \|p\bullet\| \leq 1$. P_{LS} is the set of places belonging to the LS subnet.
- **Split-OR Subnet (S-OR):** A S-OR subnet is a Petri Net which that satisfies the condition. $\|P_{S\text{-}OR}\| = 1$. If $p \in P_{S\text{-}OR}$ then $\|p\bullet\| > 1$. $P_{S\text{-}OR}$ is the set of places of a S-OR.
- **Join-OR Subnet (J-OR):** A J-OR subnet is a Petri Net which that satisfies the condition. $\|P_{J\text{-}OR}\| = 1$. If $p \in P_{J\text{-}OR}$ then $\|\bullet p\| > 1$. $P_{J\text{-}OR}$ is the set of places of a J-OR.



Notes: Some places have been duplicated for clarity. Numbers inside places indicate the process time

Figure 4. TCPN (left) and its unfolding into a TPN (right)

A Sequence of Activities (SA) is generated by properly linking these constructs. Then resource places and their corresponding arcs are added using the SSAR construction rules. In addition to such rules we constrain the operational places whose processing time is greater than zero to only allow one token at a time. Formally this is: if p $\in P_{SSA} \wedge \tau(p) \geq 0$ then $M(p) \leq 1$. This constraint guarantees that the remaining process time is not ambiguous. In the case of modeling a TCPN, there is only one start place that is initially marked with as many classes of part color tokens. The example shown in figure 4 illustrates the unfolding of the TCPN into a TPN. The TCPN on the left contains two distinct part tokens and two distinct machine tokens. This hierarchical construction has the property that a SSAR at the TCPN layer, unfolds into a SSAR at the TPN layer.

3. State Equations.

The state equations used here were subject of previous work by [6]. The main feature is the augmentation of the conventional incidence matrix and place marking with the remaining process time of for places. These equations are valid for the TPN layer. The augmented state space representation can be written as:

$$X(k+1) = A(k)\,X(k) + B(k)u(k)$$

Where: $X(k)$ is the state vector.

$$X(k) = \begin{vmatrix} M_p(k) \\ M_r(k) \end{vmatrix}$$

$M_p(k)$ and $M_r(k)$ are $n \times 1$ vectors where n is the number of discolored places of the net. In these equations the number of discolored transitions is m. $M_p(k)$ is the marking vector after k transition firings. $M_r(k)$ is the remaining processing time vector after k transition firings.

$A(k)$ is the system matrix. This matrix is partitioned as follows:

$$A(k) = \begin{vmatrix} \mathbf{I} & \mathbf{0} \\ -\tau(k)\mathbf{P} & \mathbf{I} \end{vmatrix}$$

I : Identity n x n matrix; **0:** Zero n x n matrix; $\tau(k)$ Time elapse between two consecutive transition firings.

P is a diagonal $n \times n$ matrix that serves to distinguish operational places from resource places. p_{ii} = element of **P**: 1 if the *ith* place in M_p is an operational place; 0 otherwise} $p_{ij} = 0$ when i≠j

$u(k)$ $m \times 1$ Control vector that determines which transitions fire at time k. $u_j(k)$ is defined as the *jth* position of u at time k. $u_j(k)$ = {1 if transition j fires, 0 if it does not }

$B(k)$ is the distribution matrix that transforms the control action $u(k)$ into addition or removal of tokens when firing a transition represented in vector $u(k)$.

$$B(k) = \begin{vmatrix} L \\ W L^{+} \end{vmatrix}$$

L= $n \times m$ Incidence matrix. n and m are the number of discolored places and transitions respectively.

$\mathbf{L} = \mathbf{L}^{+} - \mathbf{L}^{-}$. $\mathbf{L}^{+}$ = Incidence output matrix that accounts for the addition of tokens in output places.

$\mathbf{L}^{-}$ = Incidence input matrix that accounts for the removal of tokens from input places.

W = Processing time n x n diagonal matrix. w_{ii} = process time of the *ith* place in M_p. $w_{ij} = 0$ when i≠j

Intuitively the remaining process time is calculated as follows: At event k tokens arrive to some operational places. The remaining process time for these tokens is the process time itself since the corresponding operations have not started yet. The process time of the incoming tokens is represented by the vector $WL^{+}u(k)$. Thus the remaining process time vector $M_r(k)$ is updated with the addition of the process time of the just-arrived tokens. This yields $M_r(k)^{+} = M_r(k)^{-} + WL^{+}u(k)$ where $M_r(k)^{-}$ and $M_r(k)^{+}$ are respectively the remaining process time vectors before and after the addition of the newly arrived tokens. The firing of the next transition is determined by $\tau(k)$ which is the maximum of the remaining process time in $M_r(k)^{+}$ of its input places. The new $M_r(k+1)$ vector is recalculated by subtracting $\tau(k)$ time units from the $M_r(k)^{+}$ vector. In the state space equations this is represented by $M_r(k+1) = M_r(k)^{+} - \tau(k)\mathbf{P}\,M_p(k)$. Since at most one token can be located in the operational places, there is no confusion as to which token we are referring to. The control logic must control the flow of tokens from and to operational places thus meeting the above constraint.

4. Applications of the State Equations: Heuristic Search

Current research is focused to optimization of the execution of the net. In particular the state equations (TPN layer) are amenable with AI techniques such as A* search. A* search seeks to expand only the most promising branches of the net reachability tree based on the actual cost (g(M)) from M0 to the marking M and an estimate of the cost from marking M to the final marking *(h(M))*. The reader is referred to [5] for a detailed explanation of this methodology

applied to Timed Petri Nets. To the basic methodology we add a main feature: We explore only markings whose depth in the reachability tree are contained in a "moving window" [7]. This is if the depth of generated marking falls outside the lower and upper depths, the marking is not further expanded unless there are no more markings available. When the number of markings in the window exceeds a number (we say the window is full), the windows moves up allowing markings further deep in the reachability graph. (see [7] for a more detailed explanation). In addition we the following heuristic $h_a(M)$ to calculate the estimated cost:

$$h_a(M) = \sum TC(M)_i / NR$$

$TLC(M)_i$ is Time Left to Completion of a part token i and NR is the number of resources available. Some results are presented in table 1 using the unfolded net of figure 4. The heuristics $h=0$, $h=-2d$ and $h=-3d$, where d is the depth of the marking in the reachability graph, were used previously by [5]. In the results shown below we used a size of window =10 and maximum markings in window = 15.

Batch size = 1				
Heuristic	$h=h_a$	$h=0$	$h=-2d$	$h=-3d$
Time	34	34	35	38
Iterations	76	84	53	22

Batch size = 1				
Heuristic	$h=h_a$	$h=0$	$h=-2d$	$h=-3d$
Time	73	73	73	73
Iterations	169	161	135	109

Batch size = 5				
Heuristic	$h=h_a$	$h=0$	$h=-2d$	$h=-3d$
Time	164	165	169	169
Iterations	462	469	409	397

Batch size = 10				
Heuristic	$h=h_a$	$h=0$	$h=-2d$	$h=-3d$
Time	323	326	328	327
Iterations	942	931	877	877

5. Conclusions

This paper has presented a Petri Net modeling approach for flexible manufacturing systems which takes advantage of hierarchical task decomposition. Our modeling approach considered different nets to represent different levels of complexity. Alternative sequences and resources were incorporated in the process task model to provide flexible operation instructions. Dynamic equations and their applications in sequencing optimization were also discussed. Further research is focused towards the improvement of the heuristic functions and optimization under stochastic scenarios.

References

1. Albus, J. McLean, C. Barbara, A. Fitzgerald, M. 1983. "Hierarchical Control for Robots in an Automated Factory". 13th ISIR/ Robots Symposium.
2. Ezpeleta, J. Colom, M. Martínez, J. ". 1995. IEEE Transactions on Robotics and Automation. 11(2) 173-184
3. .Jeng, M, D. Chen, S. C. 1999. Heuristic Search Based on Petri Nets for FMS Scheduling. IEEE Transactions on Industry Applications. 35(1) 196-202.
4. Jensen, K. 1981. "Coloured Petri Nets and the Invariant Method". Theoretical Computer Science, 14(1) 317-336.
5. Lee, D. Y. DiCesare, F. 1994. "Scheduling Flexible Manufacturing Systems using Petri Nets and Heuristic Search". IEEE Transactions on Robotics and Automation. 10(2)123-131.
6. Liu, C, Ma, Y. Odrey, N. 1997. "Hierarchical Petri Net Modeling for System Dynamics and Control of Manufacturing Systems". Proceedings of the 7th FAIM International Conference. June 25-27. Middlesbrough, UK. 1997.
7. Moro, A. Yu, H. Kelleher, G. "Advanced Scheduling Methodologies for Flexible Manufacturing Systems using Petri Nets and Heuristic Search". Proceedings of the 2000 IEEE International Conference on Robotics and Automation. San Francisco, California, USA. April 2000. 2398-2403
8. Odrey, N. Ma, Y. 1995 "Intelligent Workstation Control: An Approach to Error Recovery in Manufacturing Operations". Proceedings of the 5th International FAIM Conference. Stuttgart, Germany. 124-141
9. Odrey, N. Ma, Y. 2001. "A Multi-Level Multi-Layer Petri Net Based Approach for Manufacturing Systems Control". Proceedings of the 11th FAIM International Conference. July 16-18, 200. Dublin, Ireland.
10. Wyns, J. Valckernaers, P. Van Brussel, H. Bongaerts, L. 1996 "Implementation of Resource Allocation in the Holonic Workstation Architecture". 1996. Proceedings of the 1st Europe-Asia Congress on Mechatronics. October 1-3. Besancon, France.
11. Zhou, M. DiCesare, F. 1993. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers. USA.