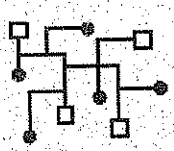


ISE

Industrial and Systems Engineering

# Modeling and Optimization of Production Systems With Timed Color Petri Nets and Heuristic Search

**Gonzalo Mejia**
**and**
**Nicholas G. Odrey**
**Lehigh University**

**Report No. 02W-001**

LEHIGH
University

200 West Packer Avenue
Bethlehem, PA 18015

# Modeling and Optimization of Production Systems with Timed Color Petri Nets and Heuristic Search

**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 describe our approach by modeling the activities of the so termed workstation level using Timed Color Petri Nets (TCPN) at the highest layer and Timed Petri Nets at the second layer. We concentrate our efforts at the lowest layer where time critical analysis is performed. For such analysis we use the heuristic A* search that has shown promising results. To improve the performance of the A* search algorithm we selectively limit the markings that can be explored. We test our algorithm on our modeling approach and also compare it with previous work. Our results show significant improvements over past work.

**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.

A 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 because they suffer of 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 adopt a classical heuristic search method in combination with features that limit the search space and a new heuristic function. Preliminary results show significant improvements over previous work.

## 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 describe the construction rules for such nets.

### 2.1 Petri Nets Definitions and Modeling Techniques:

Timed Colored Petri Net (TCPN): A TCPN is 7-tuple G = (P, T, I, O, $M_0$, $\tau$, 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}.$\tau$ = Set of time delays associated with places. C = set of colors.

#### 2.1.1 Definition: Sequence of Activities (SA)

A SA is a Petri Subnet that is intended to model the activities at a given 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 (termed here as start place) and ends in another unique place (termed here as end place). Places that belong to a SA are denoted as operational places. Places representing resources or other ways of coordination are not accounted for. The properties of a SA are:

- A SA is a connected state machine. This is each transition has only one input and one output arc.
- A SA contains no cycles.
- All places have at least one input and one output arc except the start place (no input arcs) and the end place (no output arcs). An example is shown in figure1.

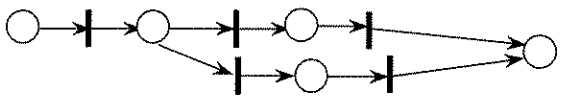


**Figure 1 A valid SA.**

#### 2.1.2 Definition: Set of Sequences of Activities (SSA).

SSA is an extension of SA's. SSA is the Petri Net formed by several SA's. An SSA consists of one or more unconnected SA's. A SSA represents the activities of all current jobs that have been allocated to an individual entity of a particular level within the manufacturing system. Usually the start place of each SA that belongs to the SSA is initially marked and the goal is to bring tokens from the start places to the end places of each SA.

### 2.1.3 Definition: Set of Activities with Resources (SSAR)

A SSAR represents the addition of resources to a set of sequences of activities. The addition of resources implies addition of places (termed as resource places) which represent, the availability of the resource, and arcs but not additional transitions. See figure 2 for an example. Resources are allocated to the SSA by connecting arcs from/to the resource places from/to the existing transitions. In our methodology we take provisions to avoid ensure the proper allocation and release of the resources. Provisions include checking that a resource is allocated and released in each branch of a SA and that a resource cannot be allocated (resp. released) twice before being released (resp. allocated. This ensures that the net remains bounded. This definition was adapted from [2].

### 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 (e.g [10]) is adopted. To avoid deadlocks we determine a control policy which maintains the net liveness using heuristic search as explained later in this document. Since in real-life applications, the control policy may not be applied exactly as planned, we are investigating techniques to maneuver out of deadlock states.

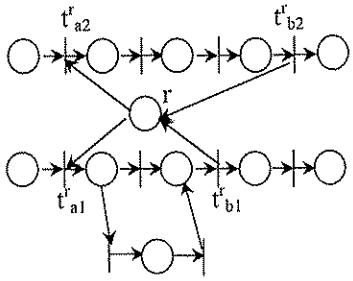



**Figure 2 a valid SSAR.**

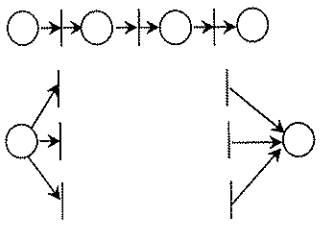


**Figure 3. (a) Linear sequence (b) a Split-OR (c) Join-OR**

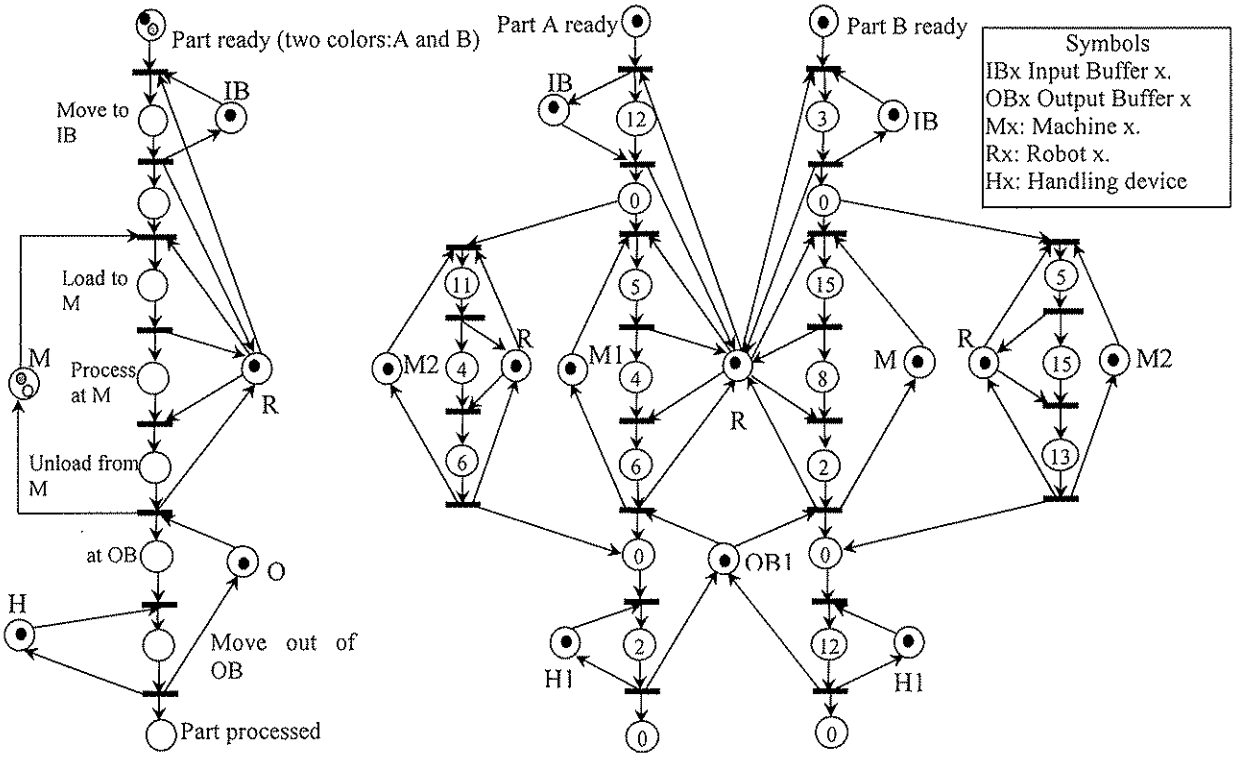
### 2.2 Petri Net Synthesis

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 [12]. See figure 3.

- Linear Sequence (LS):
- Split-OR Subnet (S-OR):
- Join-OR Subnet (J-OR):

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. 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.




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)**

## 4. Optimization using Heuristic Search.

Current research is focused to optimization of the execution of the net. In particular the TPN layer is 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 heuristic function $f(M) = g(M) + h(M)$. $g(M)$ actual time from $M_0$ to the marking $M$ and $h(M)$ is an estimate of the cost from marking $M$ to the final marking. A* uses two lists, namely OPEN and CLOSE. The list OPEN contains markings generated but not yet explored. The OPEN list is sorted in according to the heuristic function $f(M)$ and the first marking is removed from OPEN and put on the back of CLOSED. The children of the last marking of CLOSE are put on OPEN and the algorithm goes on until the final marking is reached or OPEN becomes empty. The reader is referred to [5] for further details of this methodology applied to Timed Petri Nets.

To guarantee that A* search finds an optimal solution $h(M)$ must be equal or less than the actual value $h^*(M)$ at all markings [5]. However employing admissible heuristic functions leads to breath-first strategies that produce exponential growth of markings [7]. To overcome this drawback the most common strategies are (i) limiting the scope of selection and (i) limiting the back-tracking capability of A* search [7]. Limiting the back-tracking capability of the search results in reduced complexity at the expense of optimality.

We limit scope of selection with a heuristic function $h(M)$ described later in this paper. To limit the back-tracking capability we adapt the strategy of a "moving window" [7] which excludes both too shallow and too deep markings from further expansion. With this method we explore only markings whose depth in the reachability tree are contained between predetermined bottom and top depths. Figure 5 shows an example of the moving window method applied to a net reachability graph. In our method, the first marking on OPEN is moved to CLOSE for exploration only if its depth in the reachability tree falls inside the current top and bottom depths. If no marking on OPEN falls in the window, then the first marking on OPEN is moved to CLOSE. When the number of markings in the window

exceeds a pre-defined number (we say the window is full), the windows "moves up" allowing markings further deep in the reachability graph to be expanded. When the window is full and moved up, we can adopt a strategy that might vary from (i) completely emptying the window to (ii) removing just one marking from the window. If the window is emptied some non-promising markings already discarded may be expanded. If just one or few markings were removed from the window the search would go too quickly towards markings deep in the reachability graph. Best results were obtained when the window was half-emptied. We name our method as Advanced Moving Window (AMW) algorithm. The prototype for this algorithm is AMW (*h*, *mw*, *ws*) where *h* is the heuristic function used, *mw* is the maximum number of markings in the window and *ws* is the window size.

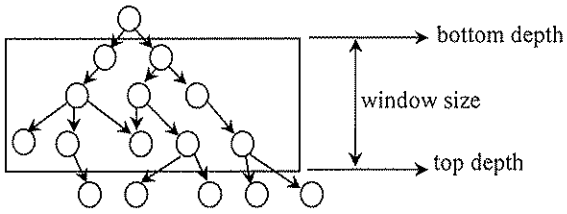



**Figure 5. The search window**

In this section we first describe our methodology and compare it to previous work and then we test our algorithm on the unfolded net shown in figure 4.

**Example 1:** To compare our results with previous work, we used the benchmark presented by [7] and [11]. The results obtained were computed from a job shop with 3 machines and 4 jobs with no alternative routings and 3 operations per job (1 per machine). We tested two heuristic functions: The first heuristic is $h_w$=-*wd*, where *d* is the depth of the marking in the reachability graph and *w* is a weight. This heuristic was used previously by [5]. The second heuristic $h_a(M)$ is defined as:

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

$TLC(M)_i$ is Time Left to Completion of a part token *i*. A token and *NR* is the total number of resources.

Time Left to Completion is addition of the optimal time that a part token requires to reach its end place plus the remaining process time at its present location. The optimal time is the time according to the process plan (or along the optimal path in case of alternative process plans). This ideal situation would arise if there were unlimited resources and tokens spent no time at all waiting in queues. The number of resources *NR* accounts for how many jobs can be performed simultaneously. The lot size for each job, the makespan and the strategy used are shown table 1.

**Table 1. Comparison Results versus [7] and [11]**

| Lot Size | *BF* | | *DWS* | | *AMW ($h_a$, 15, 20)* | | *AMW (-2d, 2,3)* | |
|---|---|---|---|---|---|---|---|---|
| *Job 1, 2, 3, 4* | MK | #IT | MK | #IT | MK | #IT | MK | #IT |
| 5, 5, 2, 2 | 58 | 3437 | 58 | 431 | 58 | 165* | 60 | 164 |
| 8, 8, 4, 4 | 100 | 9438 | 100 | 856 | 100 | 1232 | 100 | 284 |
| 10, 10, 6, 6 | 134 | 23092 | 134 | 1204 | 146 | 1577 | 134 | 380 |

Symbols: BF: Best First [11]. DWS (Dynamic Window Search) [7]. MK (Makespan). #IT. Number of Iterations
* The strategy *(ha, 2,3)* was used here.

**Example 2:** Next we test our AMW algorithm on using the unfolded net of figure 4 with different batch sizes and different strategies. The processing times are shown in figure 4. A sample of the results is shown in table 2.

**Table 2. Results of AMW with different heuristics, window sizes and window capacities.**

| Lot Size | *($h_a$, 3, 2)* | | *(-2d, 3, 2)* | | *($h_a$, 15, 20)* | | *(-2d, 15, 20)* | | *($h_a$, 2,3)* | | *(-2d, 2,3)* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MK | #IT | MK | #IT | MK | #IT | MK | #IT | MK | #IT | MK | #IT |
| 1 | 60 | 44 | 65 | 44 | 52 | 76 | 53 | 70 | 60 | 28 | 65 | 28 |
| 5 | 245 | 237 | 253 | 236 | 244 | 459 | 245 | 452 | 266 | 156 | 266 | 159 |
| 8 | 389 | 381 | 397 | 380 | 388 | 751 | 391 | 738 | 417 | 255 | 417 | 255 |
| 10 | 485 | 477 | 493 | 476 | 491 | 910 | 493 | 880 | 513 | 319 | 513 | 316 |

MK (Makespan). #IT. Number of Iterations

The results in table 2 show that the size of the window and the maximum number of markings have greater impact on the search than the heuristic function. Having small window sizes and window capacities brings quickly the search deep into the reachability graph and towards the final marking, at the expense of optimality. In the opposite case, the search can perform a broader scan on branches within the window.

In example 1 the best heuristic was $h_w$ as it can deducted from the results. However, it might not be the case when the process times are very dissimilar as shown in example 2. We tried different weights $w$ for the function without significant improvements. Even $h_a$ does not consistently outperform $h_w$, it avoids adjusting the weight $w$ of the function $h_w$. This is because $h_a$ is based on the inherent characteristics of the net.

## 5. Conclusions

This paper presents a Petri Net modeling approach for flexible manufacturing systems which takes advantage of hierarchical task decomposition. Our modeling approach considers different nets to represent different levels of complexity. Alternative sequences and resources are incorporated in the process task model to provide flexible operation instructions. To optimize the performance at the workstation level we use a modified version of the A* search algorithm which shows significant improvement over previous work. 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. Xiong, H. Zhou, M. 1998. IEEE Transactions on Semiconductor Manufacturing. v11(3) pp384-393
12. Zhou, M. DiCesare, F. 1993. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers. USA.