K.U.Leuven

Department of Electrical Engineering (ESAT) **SISTA**

Technical report 95-63

Methodologies for discrete event dynamic systems: A survey*

L. Ben-Naoum, R. Boel, L. Bongaerts, B. De Schutter, Y. Peng,
P. Valckenaers, J. Vandewalle, and V. Wertz

If you want to cite this report, please use the following reference instead:
L. Ben-Naoum, R. Boel, L. Bongaerts, B. De Schutter, Y. Peng, P. Valckenaers, J. Vandewalle, and V. Wertz, "Methodologies for discrete event dynamic systems: A survey," *Journal A*, vol. 36, no. 4, pp. 3–14, Dec. 1995.

ESAT-SISTA
K.U.Leuven
Leuven, Belgium
Current URL: https://www.esat.kuleuven.be/stadius

Methodologies for Discrete Event Dynamic Systems: A Survey*

Lamia Ben-Naoum[†] Rene Boel[‡§] Luc Bongaerts[¶‖] Bart De Schutter[**††]
Yanming Peng[¶] Paul Valckenaers[¶] Joos Vandewalle[**] Vincent Wertz[†§]

Summary

In recent years both industry and the academic world are becoming more and more interested in techniques to model, analyse and control complex systems such as flexible production systems, parallel processing systems, railway traffic networks and so on. This kind of systems are typical examples of discrete event dynamic systems, the subject of an emerging discipline in systems and control theory. In this paper we present an overview of some of the important methodologies for discrete event dynamic systems. All these methods have been studied intensively by the authors in the framework of projects.

1 Introduction

A discrete event dynamic system (DEDS) is a dynamic, asynchronous system, where the state transitions are initiated by events that occur at discrete instants of time. Typical examples of DEDS are flexible manufacturing systems, telecommunication networks, traffic control systems, multiprocessor operating systems, ... An event corresponds to the start or the end of an activity. In the case of a production system possible events are: the completion of a part on a machine, a machine breakdown, a buffer becoming empty, and so on. Intervals between events are not necessarily identical; they can be deterministic or stochastic. DEDS typically exhibit an asynchronous behaviour with a lot of parallelism and interaction with their environment. DEDS are usually man-made and are characterised by a complex, hierarchical structure. In recent years much attention has been devoted to DEDS both in academic research and industrial development in order to guarantee safety and to improve efficiency and flexibility.

There are many frameworks to model, analyse and control DEDS. In this paper we present a survey of the most important DEDS methodologies studied by the authors. In Section 2

* This paper presents research results of the Belgian programme on interuniversity attraction poles (IUAP-17 and IUAP-50) initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture. The scientific responsibility is assumed by its authors.

[†]CESAME, Université Catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium.

[‡]Vakgroep Elektrische Energietechniek, University of Ghent, B-9052 Gent, Belgium.

[§]Research associate with the N.F.W.O./F.N.R.S. (Belgian National Fund for Scientific Research).

[¶]PMA, K.U.Leuven, B-3001 Heverlee (Leuven), Belgium.

[‖]Research Assistant supported by the I.W.T. (Flemish Institute for Support of Scientific-Technological Research in Industry).

[**]ESAT/SISTA, K.U.Leuven, B-3001 Heverlee (Leuven), Belgium.

[††]Research assistant with the N.F.W.O.

we first describe why one would want to model, analyse and control DEDS. In the next sections we give an overview of Petri nets, controlled Petri nets, max algebra and Lagrangian relaxation. We also give a brief explanation of the connection between Petri nets, Grafcets and PLC design. We conclude with a comparison of the advantages and disadvantages of the methodologies mentioned above.

2 Goals of DEDS methodologies

In this section we briefly discuss three major objectives in DEDS research: modelling, analysis and control.

2.1 Modelling and analysis

When we want to (re)design a system or develop a controller to ensure that the system meets certain requirements, or when we want to verify or optimise its behaviour, we should represent our knowledge about the properties and the behaviour of the system in a model allowing prediction of the performance of the system. There are many modelling and analysis techniques for DEDS, such as queueing theory, (extended) state machines, max algebra, formal languages, automata, temporal logic, generalised semi-Markov processes, Petri nets, perturbation analysis, computer simulation and so on. In this paper we shall only discuss some of these methods. For more information on the other methods the interested reader is referred to [8, 19, 20, 27, 36] and the references cited therein. All these techniques have their advantages and disadvantages and it really depends on the system we want to model and on the goals we want to achieve which of the above methodologies best suits our needs. When we have to select the most appropriate method, then one of the most important trade-offs that has to be taken into account is modelling power versus decision power: the more accurate the model is the less we can analytically say about its properties.

Up to now the most widely used technique to study DEDS is certainly computer simulation. One of the major disadvantages of computer simulation is that it is computationally quite demanding since it requires a high degree of detail in the model. However, this also leads to a high degree of correspondence between the model and the real system. Another disadvantage of computer simulation is that it does not always give us a real understanding of the effects of parameter changes on properties such as robustness, stability, optimality of the system performance (did we reach a global or a local optimum?), and so on. Hence, we prefer to use mathematical models that are more suited for mathematical analysis and that allow us to make predictions about the system behaviour and to analytically design good controllers. The main appeal of a mathematical description of the system lies in the existence of efficient algorithms to evaluate system performance. This is a significant advantage over time-consuming and expensive simulation, which is usually required to obtain the same information. Furthermore, analytic techniques also provide a better insight in the effects of parameter changes on the system properties. However, as we have already mentioned, we have to take into account that there is a trade-off between the accuracy of our model on the one hand and the techniques available for analysis on the other hand. Therefore, we normally use a mixture of analytic methods, approximations and computer simulation when we want to study the properties and the behaviour of DEDS.

2.2 Control and optimisation of system performance

The aim of DEDS control is to develop controllers and control strategies that improve the global performance of the system. Hereby we consider both qualitative properties (safety, prevention of buffer overflow, prevention of congestion, prevention of deadlock, ...) and quantitative properties (maximal throughput, minimal average delay, minimal stocks, optimal employment of resources, ...).

The first aim of DEDS control is to ensure that the system behaves 'properly': dangerous situations must be avoided, the system should not reach a state from which it cannot leave (deadlock) and so on. In order to control the system the controller must be able to prevent certain transitions from occurring: e.g. cars can be prevented from entering an intersection by setting a traffic light to the position 'red'. Transitions that can be blocked are called controllable transitions. The decision on which controllable transitions should be blocked from happening will be based on knowledge about the past behaviour of the system. Hence, at each time, the controller should have at its disposal certain partial observations of the evolution of the system state. Using the model specification the ensemble of all possible future event sequences can be calculated. In particular it is possible to determine whether the set of possible future event sequences contains any undesirable event sequences. If there are such undesirable sequences, then one has to determine how they can be prevented by blocking as few controllable transitions as possible. It is always important to find a control that blocks as few transitions as possible, since this allows maximal freedom to the evolution of the system and since this allows a maximal degree of freedom for a higher level controller that has to optimise other properties of the system. However, even for small-sized systems the number of possible future event sequences can explode combinatorially. This means that in general the system model (and thus also the control design) would become hopelessly more complicated if one described the full behaviour of the system. Therefore, it is necessary to have a compact but also sufficiently accurate mathematical description of the system, which also allows an efficient search for a good control strategy. By efficient search, we mean a procedure that does not depend on the enumeration of all the possibilities one by one.

Note that the fundamental idea for continuous system controllers (such as PID, LQG, H_∞) and for DEDS controllers is the same: define a partitioning of the set of all possible input-output sequences in a subset of 'good' sequences and its complement the subset of unacceptable sequences. In the case of a PID controller this is expressed in terms of the long term behaviour (stability), the overshoot, behaviour of the high frequency components, etc., or in terms of the cost associated with trajectories for a particular choice of state model (but note that state models for a given input output behaviour are not unique). For DEDS the same is true. We want to design a control guaranteeing that certain sequences of transitions will never occur. This may for example be specified via a set of forbidden states for a particular Petri net (state space) model for the DEDS. The set of forbidden states may be chosen in such a way that even in the case of failure of some components of the system, safety will not be affected. Afterwards it is possible to try to schedule events in such a way as to achieve good performance, under the safety constraints imposed by the first layer of control. What is different between continuous time systems and DEDS is that the control values chosen for a continuous system are typically a small number of real values (settings for adjustable parameters), while in a DEDS the control is specified by a large number of binary values (enabling or disabling transitions).

As an example of a forbidden state specification consider the problem of deadlock avoid-

ance. Deadlock occurs when several processes are each holding a number of resources, and no process can proceed to completion because each process is waiting for the other processes to be completed. For example material can be removed from a buffer only when another buffer or another resource is free, but then this resource can only be freed by placing the workpiece it is working on in this same buffer. This leads to deadlock whenever there is a cycle, a circular chain, of such interdependent resources and buffers. Deadlock is avoided by making sure that the state where each of the resources and buffers of the cycle are full can never occur.

So far we have only considered the logical behaviour of the DEDS, i.e. the sequence in which the events take place without taking into account the time between two consecutive events. In most applications one is also interested in optimising quantitative properties of the system, such as producing as many items as possible in a given time span, having as many cars as possible flow through an intersection, etc. This is the second aim of DEDS control. The first level controller specifies a control that allows as large a set of possible sequences of future events to take place as is compatible with the safety constraints. This usually still leaves a large degree of freedom in choosing the actual order in which events are executed. This freedom can be used to optimise quantitative measures: throughput, delays, tardiness, and so on. This will typically result in a scheduling problem in which we have to find a strategy that determines e.g. which jobs have to be processed on which machines, the optimal order to process the jobs and how we have to adapt the schedule in case of a rush order or a (minor) system failure.

3 Petri nets — synthesis, simulation and control

3.1 An introduction to Petri nets

An interesting class of mathematical models for discrete event systems are Petri nets. Petri nets give a graphical representation of a DEDS, which is a mathematical model very closely resembling the physical reality. At the same time the Petri net formalism is very well suited for implementation as a computer language.

An example of a Petri net, representing an assembly plant, is shown in Figure 1. Petri nets consist of a number of places — drawn as circles — interconnected via directed arcs to transitions — drawn as bars. Places can be marked by tokens — indicated graphically by marking the circle with a point. The number of tokens in p is indicated as $m(p)$. The set of all places from which an arc starts, pointing towards a given transition t, forms the set of input places of transition t. If all input places of transition t are marked by at least one token, then transition t is enabled and can fire. When t fires, it removes one token from each of its input places, and puts one token in each of its output places — the places reached by an arc that starts at the given transition t. A Petri net is said to be in deadlock if none of the transitions is enabled. The state of the Petri net is described by specifying the number of tokens in each of the places of the net. With such a model of a DEDS, it is usually simple to describe unsafe situations or deadlocks as particular sets of markings of the Petri net. If the tokens represent workpieces held in the buffer or being processed by the resource, deadlock would occur if all the places of a cycle simultaneously hold the maximum number of tokens they can store. This is an example of a forbidden state specification [2, 6].

The basic ingredients of a complex discrete event system are synchronisation between certain events and choice between certain event sequences. In a Petri net choice is modelled by having a place with more than one output transition. Either of these output transitions

can fire if the place is marked by one token, but not more than one of these transitions can fire (using the same token). The place p_6 in Figure 1 is an example of choice: either transition t_1 or transition t_3 can fire. This means that the robot can be used for loading, processing and unloading a part either on workstation W_1 or on workstation W_3, but not on both at the same time.

In an assembly plant it is essential that the task of assembling several pieces together only starts after all the pieces have become ready and available. This kind of synchronisation requirement is modelled by having a transition with more than one input place. The transition can only fire if all the input places are marked simultaneously. The transitions t_1, t_2 and t_3 in Figure 1 are all examples of synchronisation. For t_1 and t_3 the synchronisation means that both a part and a robot have to be available in order to proceed with the operation. For t_2 it means that there has to be an empty place in the buffer in W_2 to hold the part in order to start with the operation in W_2. Another form of synchronisation occurs when several places always become marked simultaneously. This can be modelled by a transition with several output places. It occurs e.g. when a piece enters a buffer and at the same time the robot that operated on the piece becomes available again.

3.2 Using Petri nets to model flexible manufacturing systems

Flexible manufacturing systems (FMS) are typical examples of asynchronous concurrent systems. Petri nets have evolved into a powerful tool for analysing asynchronous concurrent systems. To model a complex manufacturing system using Petri nets, deadlock prevention is one of the key issues. Two different methods, deadlock prevention and deadlock avoidance, will be explored in this and the next section.

The first method, which will be discussed in this section, uses standard Petri nets to avoid possible deadlock by carefully constructing the Petri net model of the system [34, 39, 40, 41]. The Petri net model includes all deadlock prevention information as well as other information (resource sharing, precedence constraints, etc.). So in this case the deadlock prevention mechanism is also modelled with Petri nets.

The second method uses controlled transitions to prevent possible deadlock [3, 24, 28]. Control places are added to some of the transitions in order to prevent them from being fired in certain circumstances. This will be described in more detail in Section 3.3.

These methods have been applied to the pilot flexible assembly system that was built at the division PMA of the K.U.Leuven (Belgium). A description of this testbed (See Figure 2) can be found in [38]. Using Petri nets, an integrated simulation/control system for the flexible assembly system has been designed and implemented [34, 39]. It can serve as a simulator or a real-time controller. The system functions as a simulation when feedback information is generated from a random number generator and some stochastic distribution functions. Currently, the research activities have only tested the simulation/controller system in its simulation mode. The system functions as a control system when the parts of the Petri net model that correspond to the underlying physical production system are made to reflect the real-world situation. For instance, when a pallet enters a buffer, the corresponding transition in the Petri net will be fired. Conversely, the Petri net interpreter will issue commands to the real-world system when it fires a corresponding transition in the model (e.g. to start executing an assembly operation). Evidently, such a link between the Petri net simulation/controller and the real-world system requires a significant software implementation effort, which today must be repeated for every new physical production system.

The control of the flexible assembly system is partitioned into 2 smaller subproblems: the first one deals with resource sharing and the precedence constraints of the products. The second one deals with the automatic transport system. Two Petri net models are generated automatically for each of the subproblems. A simulation engine, which is the kernel of the integrated simulation/control system, takes those two Petri net models as input and synchronises/co-ordinates them in the time domain.

Modelling

Four important types of information are presented by Petri net models: the precedence graphs of the products, the shared manufacturing resources, the product routes, and the deadlock prevention mechanisms. The first two are not difficult to model. A coloured Petri net [25], where tokens with different colours represent different products, can be used to model the various product routes. Modelling the deadlock prevention mechanisms is the most difficult task. The Petri nets that are automatically generated are guaranteed to be deadlock-free. This is achieved through the design of suitable Petri net building blocks and formal proof that their combinations, which are generated automatically, are indeed deadlock-free; details [34, 39] are beyond the scope of this paper. Neither the design of these building blocks nor the construction of the formal proof happens automatically; it requires a properly trained human being. However, the set of Petri net models that can be obtained through the automatic combination of the building blocks represents a useful collection for the application domain, and allows one to guarantee that the resulting models are deadlock-free even when they become very complex/large. In summary, this approach is not fully automated but is capable of handling problems of more realistic size and complexity than most known research results in the domain.

Timed Petri nets [16, 35] are used for both the product/FMS Petri net and the transport Petri net model. In timed Petri nets times are assigned to transitions. The execution of a transition takes a certain time that can be stochastic or deterministic. During the firing, the enabling tokens of the transition are reserved and not available for other transitions. There is no 'overtake' on a timed transition: the firing transition cannot be fired again.

Priority Petri nets [35, 37] are used for solving scheduling problems. In priority Petri nets numbers (priorities) are assigned to the transitions. When more than one transition is enabled, the one with the highest priority will be selected to fire. Implementing different (heuristic) dispatching rules is equivalent to writing a compare function that determines which one of two given transitions has higher priority.

A Petri net model of the transport system is generated automatically according to the layout of the automatic transport system of the flexible assembly system. The initial marking of the transport Petri net model represents the initial position of the pallets in the transport system.

Simulation and control

The control of the flexible assembly system is partitioned into 2 smaller subproblems: the scheduler and the transport controller. The scheduler takes the precedence graph and the resource requirements of the products as input and proposes a least commitment schedule to the controller. The transport controller takes the proposed schedule as input and makes the final decision based on the status of the transport system. It tries to minimise the transport

time and always honours the (partially ordered) schedule proposed by the scheduler. Both the scheduler and the transport controller make decisions opportunistically using heuristic dispatching rules. The Petri net models guarantee that all the constraints are respected regardless of what scheduling and control strategies are used. Different scheduling/controlling strategies can be implemented for the simulation engine. Heuristic dispatching rules are used for the scheduling and the implementation is simple and straightforward. This is the major advantage of using Petri nets in this way to handle scheduling/control of FMS. Another advantage of this method is that one can add more products dynamically, which provides further flexibility for the simulation.

The simulation engine is responsible for the co-ordination and communication between the scheduler and the transport controller. It integrates the two subsystems into one. It runs in two modes: simulation mode or real-time control mode. In simulation mode, the event times are calculated according to some stochastic distribution and heuristic dispatching rules are determined by the human operator. The system will simulate the behaviour of the flexible assembly system for a given scheduling/control strategy. In this mode some statistical data for the selection of the scheduling/control strategies can be gathered. This will provide useful information for the human operator to make final decisions. In real-time control mode, the event times are collected in real time from the flexible assembly system and the system will select appropriate scheduling/control strategies and control the flexible assembly system in real time.

Optimisation

There are two kinds of optimisation: the selection of scheduling/control strategies and the determination of the system parameters. There are many heuristic dispatching rules for scheduling and control of transport systems. It is difficult to find one dispatching rule that suits all situations. There always is an optimal set of dispatching rules for any given production load. The simulation provides useful information to determine 'good' dispatching rules for the given production load. Once the scheduling/control strategies are determined, there is still room to determine the system parameters such as the mix of the different products and the number of pallets per product. Different products normally have different resource requirements. The optimal mix of different products will help to balance the resource utilisation and increase the system throughput. The optimal number of pallets per product will also directly affect the system throughput.

For more information on the integrated simulation/control system for the pilot flexible assembly system the interested reader is encouraged to consult [34, 39].

3.3 Controlled Petri nets

Consider Figure 3. It is quite easy to interpret this figure as a metro network. While modern dispatching centres for a rail or metro network have very accurate information on the position of each train, we assume, in order to obtain a very robust control design, that only the fact whether a train is in a particular section of the line is known. Typically these sections are a few hundred metres long (though in switching yards smaller sections may be used). To each section we associate 4 different places in the Petri net, drawn under each other in Figure 3. One place corresponds to a train crossing the corresponding section at normal speed, another place corresponds to a train standing still or moving very slowly through the section; a third

place corresponds to a train slowing down from normal speed to a very low speed, and a fourth place corresponds to a train speeding up from slow speed to normal speed in the section. Inertia of a train has been taken into account in Figure 3 since we see that a train leaving a section at normal speed can only reach the places that correspond to passing the next section at normal speed or slowing down; similarly a train leaving a section at slow speed can only reach the places that correspond to passing the next section slowly or speeding up in the next section. Signals at the entrance of some sections force trains to slow down or stop. This corresponds to the transitions leading to the places where trains go at normal speed or where trains speed up, being prevented from occurring. One design goal could be to ensure that there are never 2 or more trains in the same section (i.e. the 4 places corresponding to 1 section cannot contain more than 1 token at the same time) and that there are at least N free sections ahead of a train running at normal speed (i.e. the $4N$ places corresponding to the N sections ahead of a train running at normal speed or speeding up should not contain any token). The dispatcher of a train can choose for each train at the entrance of each section to allow the train to run at normal speed or to slow it down (assuming the train is running at normal speed in the preceding section), provided the transition representing entering the section at normal speed is not blocked by a red light.

Controlled events correspond to transitions in the Petri net that can only fire if a control place — to be marked by the operator — contains a token. In drawings of Petri nets, such control places are indicated by rectangles. Such controlled Petri nets can be prevented from executing certain undesirable sequences of transitions by disabling some of the controllable transitions. This corresponds to not marking certain control places ('red signals' in the metro network). In the manufacturing example of Section 3.2 one designed a Petri net modelling both the evolution of the plant and the deadlock prevention algorithm. In this section we discuss an alternative approach, using a Petri net model only for the plant. Given a set of forbidden states, we construct an algorithm that works as follows: Each time a transition has been executed, the new state is observed; based on this observation, the algorithm calculates for each controlled transition whether it should be disabled in order to prevent the state from ever reaching the forbidden set. Moreover, the algorithm disables as few controlled transitions as possible, leaving maximal freedom for a higher level controller (e.g. for maximising throughput). Since this algorithm is implemented on-line on a control processor, we do not make any attempt to model it as a Petri net. Sometimes the algorithm may be representable as a Petri net, in other cases this may not be possible.

The easiest way to represent the goal of a controller for a Petri net is to describe a set of forbidden states. As a first step, consider a constraint that place p should never contain more than k tokens. We want to achieve this by blocking the smallest number of transitions possible, i.e. by impeding the flow through the network as little as possible. For a Petri net without synchronisation requirements this problem can be solved as follows. Let us call the influencing zone for the place p the set of all places contained in paths leading towards p such that these paths contain no controlled transitions and no loops. Clearly any token in the influencing zone of p can reach p no matter what control is used. Between the present position of the token and p there is no controllable transition that could be blocked to stop the progress of the token. Hence, the control design problem does not have a solution if the influencing zone, including p, already contains more than k tokens. If the influencing zone contains $l \leq k$ tokens, and if more than $k - l$ of the input transitions leading to the influencing zone are enabled by having all their input places marked, then $k - l$ of these controlled transitions at the entrance of the influencing zone will have to be blocked.

Similar algorithms work for more complicated control goals, such as upper limits on the weighted number $\sum_i \nu_i m(p_i)$ in a set of places $p_i, i \in I$, or for unions and intersections of such constraints [3, 4]. It turns out that there exist efficient algorithms for most interesting constraints, provided the forbidden sets are such that if a given state is forbidden, then any state with at least as many tokens in any place is also forbidden. The reason for this is that the control design is very safe: it can prevent a transition from firing, but it will never force a transition to fire. Thus the control design is also robust against component failures, but the model never can ensure that a certain transition will actually execute.

Similar efficient methods exist for Petri nets without choice. This has been studied in detail by Holloway and Krogh [24]. Especially in this case it is very important to consider timed models. Here one assumes that transitions take a certain amount of time to fire, i.e. there is a specification of the time interval between the moment when transition t becomes validated, and the moment it actually fires. Once this timing information is included in the model, it becomes possible to study performance measures such as throughput. If the firing times are deterministic these timed Petri nets can be represented as max-algebraic models. This will be considered in the next section.

Recently there has been done a lot of work in connecting Petri nets, Grafcets and PLC design. Grafcet [10, 11] is a standardised graphical representation of all the logical interconnections in a DEDS. Grafcets are very similar to Petri nets, except for the fact that the transitions in a Grafcet get executed immediately after they become both state enabled and control enabled (Note that in the computer literature on Petri nets as well as in the Grafcet literature one talks of an interpreted transition for what has been called a controlled transition in this paper). Thus in a Grafcet the control enabling at the same time adds a timing specification for the event represented by the transition. A promising methodology for designing logic controllers for DEDS is as follows. First take a simple Petri net model of the plant (simple because it does not include any of the specifications) and explicitly state all the specifications to be met after control has been implemented. This can be done via successive steps, first incorporating the most important specifications using the methods described in this subsection, then simulating the performance, and then, if still possible, adding less important specifications, and so on until behaviour is found satisfactory. The controlled Petri net can then be translated into a Grafcet — this is quite easy because of the close similarity between Grafcets and Petri nets. Finally transform the graphical Grafcet representation into code for a PLC that implements the control of the plant. In recent years there has been a lot of research [14, 26] on automatic translation of Grafcets into PLC code (via Relay Ladder Logic or via State Function Charts) so that this can also be an automatic operation.

4 Max algebra

Although DEDS lead to a non-linear description in linear algebra, there exists a subclass of DEDS for which the model becomes 'linear' when we formulate it in the max algebra [1, 7, 9]. The main operations of the max algebra are the maximum and the addition. We use the following notations to represent the basic operations of the max algebra:

$$\begin{aligned} a \oplus b &= \max(a, b) \\ a \otimes b &= a + b \end{aligned}$$

with $a, b \in \mathbb{R} \cup \{-\infty\}$. The reason for choosing these symbols is that a lot of concepts and properties of linear algebra (such as eigenvectors and eigenvalues, Cramer's rule, ...) can be translated to max algebra by replacing + by $\oplus$ and $\times$ by $\otimes$. This analogy is one of the most attractive features of the max algebra since it allows us to translate many concepts and techniques from linear system theory to discrete event system theory. However, there is also one major difference that prevents a straightforward translation of properties from linear algebra to max algebra: in general there exists no inverse element for the maximum operator. It should also be pointed out that the max algebra is not fully developed yet and that a lot of research on this subject is still needed in order to get a comprehensive system theory.

We shall now show how a simple production system can be modelled using max algebra. First we need some extra definitions. The operations $\oplus$ and $\otimes$ are extended to matrices in the usual way. So if A and B are m by n matrices, then $(A \oplus B)_{ij} = a_{ij} \oplus b_{ij}$ and if A is an m by p matrix and B is a p by n matrix, then $(A \otimes B)_{ij} = \bigoplus_{l=1}^{p} a_{il} \otimes b_{lj}$.

Example: A simple production system.
Consider the production system of Figure 4. This system consists of 3 processing units P_1, P_2 and P_3 and 2 buffers B_1 and B_2. Raw material is fed to P_1 and P_2, processed and sent to P_3 where assembly takes place. Between P_1 and P_3 and between P_2 and P_3 there are buffers that can each hold one intermediate part. The processing times for P_1, P_2 and P_3 are $d_1 = 9$, $d_2 = 10$ and $d_3 = 7$ time units respectively. We assume that the raw material needs 3 time units to get from the input of the system to P_1. The other transportation times are assumed to be negligible. We assume that there are buffers at the input and at the output of the system with a capacity that is large enough to ensure that no overflow will occur. Define:

- $u(k)$: time instant at which raw material is fed to the system for the $(k+1)$st time,
- $x_i(k)$: time instant at which the ith processing unit starts working for the kth time,
- $y(k)$: time instant at which the kth finished product leaves the system.

A processing unit can only start working on a new product if it has finished processing the previous one and if the buffer at its output is empty. We assume that each processing unit starts working as soon as all parts are available. Processing unit P_1 can start working on the $(k+1)$st part as soon as the following conditions are satisfied: the $(k+1)$st batch of raw material is available at the input of P_1, P_1 has finished processing the kth part, and the buffer B_1 is empty, i.e. P_3 has finished processing the $(k-1)$st part. This leads to

$$x_1(k+1) = \max(\, u(k) + 3,\, x_1(k) + 9,\, x_3(k-1) + 7) \ .$$

Using an analogous reasoning for P_2 and P_3 we get

$$\begin{aligned} x_2(k+1) &= \max(\, u(k),\, x_2(k) + 10,\, x_3(k-1) + 7) \\ x_3(k+1) &= \max(\, x_1(k+1) + 9,\, x_2(k+1) + 10,\, x_3(k) + 7\,) \ . \end{aligned}$$

In order to get a first order recursion relation we introduce an extra state variable $x_4(k) \stackrel{\text{def}}{=} x_3(k-1)$. Then we get the following evolution equations for the system:

$$x_1(k+1) = \max(\, x_1(k) + 9,\, x_4(k) + 7,\, u(k) + 3\,)$$

$$
\begin{aligned}
x_2(k+1) &= \max(\, x_2(k)+10,\, x_4(k)+7,\, u(k)\,) \\
x_3(k+1) &= \max(\, x_1(k)+9+9,\, x_4(k)+7+9,\, u(k)+3+9,\, x_2(k)+10+10, \\
&\qquad\quad x_4(k)+7+10,\, u(k)+10,\, x_3(k)+7\,) \\
&= \max(\, x_1(k)+18,\, x_2(k)+20,\, x_3(k)+7,\, x_4(k)+17,\, u(k)+12\,) \\
x_4(k+1) &= x_3(k) \\
y(k) &= x_3(k)+7\ ,
\end{aligned}
$$

or in max-algebraic matrix notation:

$$
\begin{aligned}
x(k+1) &= \begin{bmatrix} 9 & -\infty & -\infty & 7 \\ -\infty & 10 & -\infty & 7 \\ 18 & 20 & 7 & 17 \\ -\infty & -\infty & 0 & -\infty \end{bmatrix} \otimes x(k) \ \oplus \ \begin{bmatrix} 3 \\ 0 \\ 12 \\ -\infty \end{bmatrix} \otimes u(k) \\
y(k) &= \begin{bmatrix} -\infty & -\infty & 7 & -\infty \end{bmatrix} \otimes x(k)\ ,
\end{aligned}
$$

where $x(k) = [\, x_1(k)\ \ x_2(k)\ \ x_3(k)\ \ x_4(k)\,]^T$.
Note that this production system can also be represented as a timed Petri net. □

If we limit ourselves to time-invariant deterministic systems in which the sequence of the events and the duration of the activities are fixed or can be determined in advance (like repetitive production processes), we can always write down a max-algebraic state space model of the form

$$
\begin{aligned}
x(k+1) &= A \otimes x(k) \ \oplus \ B \otimes u(k) \\
y(k) &= C \otimes x(k)
\end{aligned}
$$

where $u(k)$ is the input vector, $x(k)$ the state vector and $y(k)$ the output vector. Note that this state space description closely resembles the classical state space description for linear time-invariant systems.

Once we have a model of this form, we can determine the following properties of the system rather easily:

- If we know the time instants at which raw material is fed to the system, we can calculate the time instants at which the finished products leave the system.
- If we know the latest times Y at which the finished products have to leave the system, we can calculate the time instants U at which the raw material has to be fed to the system. This amounts to solving a system of max-linear equations of the form $H \otimes U = Y$.
- However, if we have perishable goods it is sometimes better to minimise the maximal deviation between the desired and the actual finishing times. In this case there also exist an analytic expression for the optimal starting times.
- If the raw material is fed to the system at such a rate that the input buffer never becomes empty, we can calculate the steady state behaviour of the system by solving a max-algebraic eigenvalue problem of the form $A \otimes x = \lambda \otimes x$. The eigenvalue λ will be the average duration of a cycle of the production process (in steady state) and $\frac{1}{\lambda}$ is the average production rate. This also allows us to calculate the utilisation levels of the

various machines in the production process. Furthermore, the algorithm to calculate the eigenvalue also yields the critical paths of the production process and the bottleneck machines.

One of the main advantages of an analytic max-algebraic model of a DEDS is that it allows us to derive some properties of the system (in particular the steady state behaviour) fairly easily, whereas in some cases simulation might require quite a large amount of computation time.

If we allow variable or stochastic processing and transportation times and variable routing, we can still describe the system by a max-algebraic model [30, 31, 32] but it will not be time-invariant and max-linear any more, which means that it becomes more difficult to analyse its properties mathematically. For more information on the max algebra, its application to the study of DEDS and ongoing research the interested reader is referred to [1, 7, 9, 12, 13] and the references therein.

5 Lagrangian relaxation

Besides the DEDS methodologies already described, there also exists a lot of research on scheduling of discrete manufacturing. Although seen from a completely different viewpoint, the fundamental problems and the application areas are similar for both methodologies. An important difference in viewpoint with respect to the research goal is that several typical DEDS methodologies — and more specifically those that work with untimed models — focus on the safety of operation, while scheduling focuses on the performance of the system with respect to time.

Next to the more classical scheduling algorithms and heuristics found in operational research, nowadays methodologies are appearing that combine a near-optimality of the performance with a reasonable calculation time for a realistic application. An example of such a near-optimal realistic methodology is Lagrangian relaxation. Like max algebra, it is a discrete event application of methodologies that proved their value in the continuous space.

Lagrangian relaxation has been used for several types of scheduling problems. The example given here discusses the non-preemptive scheduling of multi-operation jobs with due dates on several machines ('job shop' scheduling problem). The objective of the schedule is to meet the due dates as well as possible. This corresponds to a minimisation problem for the weighted quadratic tardiness [29].

First the problem formulation is elaborated. Then the theoretical background is discussed and applied to the problem. Finally, some remaining problems are discussed. Additional information can be found in [5].

5.1 Problem formulation

First, a mathematical problem formulation of the scheduling problem is stated. For job shops, a common formulation is as follows:

$$\text{minimise } J \equiv \sum_i w_i T_i^2 \ ,$$

where J is the cost function, $T_i = \max(0, c_{iN_i} - dd_i)$ is the tardiness of order i, w_i is the weight of order i and dd_i is the due date for order i.

This minimisation problem is subject to three types of constraints: the capacity constraints, the precedence constraints and the processing time requirements. The capacity constraints are represented by

$$\sum_{i,j} \delta_{ijkh} \leq M_{kh} \qquad \text{for } k = 1, 2, \ldots, K \text{ and } h = 1, 2, \ldots, H \ ,$$

where $\delta_{ijkh} = 1$ if operation j of job i is being processed on machine type h in time period k, and $\delta_{ijkh} = 0$ otherwise; M_{kh} is the capacity of machine type h during time period k; K is the time horizon and H is the number of machine types. The precedence constraints are represented by

$$c_{ij} + S_{ijl} + 1 \leq b_{il} \qquad \text{for } i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, N_i - 1 \text{ and } l \in I_{ij} \ ,$$

where b_{ij} and c_{ij} are the beginning time and the completion time of operation j of job i; S_{ijl} is the minimum time delay between the end of operation j and the beginning of operation l of job i (e.g. caused by transportation); I_{ij} is the set of operations of job i directly following operation j of job i; N is the number of orders and N_i is the number of operations of job i. The term '+1' is introduced because c_{ij} and b_{il} refer to time periods instead of to time instances. c_{ij} is the last time period in which operation j of job i is executing, thus — apart from some delay S_{ijl} (e.g. for transportation) — operation l of job i can only start one time period later.
The processing time requirements are represented by

$$c_{ij} - b_{ij} + 1 = t_{ijm_{ij}} \qquad \text{for } i = 1, 2, \ldots, N \text{ and } j = 1, 2, \ldots, N_i \ ,$$

where $t_{ijm_{ij}}$ is the processing time of operation j of job i, if this operation is allocated to machine type m_{ij}. The term '+1' is introduced for the same reason as for the precedence constraints.

Notice that b_{ij} and m_{ij} are the decision variables. All other variables can be deduced from these ones.

The constraints can be modelled as well via the max-algebraic approach of the preceding section, and also via timed Petri nets (cf. Section 3). The ability to make decisions, however, is hard to model in the max algebra. As such, optimisation in general is hard to apply to classical discrete event system modelling techniques.

5.2 Lagrangian relaxation

The basic concept behind Lagrangian relaxation is to get rid of difficult constraints by relaxing them, and adding a penalty term in the objective function for disobeying this constraint. If the penalty term is added before the relaxation is made, it is easy to see that the solution of the modified minimisation problem is a lower bound for the optimal value of the original objective function. The higher the lower bound is, the more interesting this lower bound becomes. Fisher gives a very clear introduction to Lagrangian relaxation in [15].

Another idea behind Lagrangian relaxation is to give a weight to the penalty of each relaxed constraint. This weight is called the Lagrangian multiplier. By fine-tuning these multipliers, the lower bound can get tighter. It can be proven that for Continuous (i.e. non integer) Linear Programming Problems the lower bound is equal to the optimal cost, if the set of multipliers is chosen optimally. This property cannot be generalised for the case of integer programming problems, but the gap between the lower bound and the optimal solution often

is quite small. By relaxing the constraints, the problem is transformed from a minimisation problem in the space of the decision variables into a maximisation problem in the multiplier space.

The obtained schedule however is not a feasible one because some constraints have been relaxed. A heuristic procedure should make this schedule feasible while keeping its cost near to the optimal one. The difference between the lower bound (called dual cost) and the cost of the feasible schedule is called the duality gap. It is a measure of the quality of the schedule.

In Lagrangian relaxation, the difficult question is the choice of which constraints to relax. In the algorithms described by Luh [21, 22, 23, 29, 33], this decision is based on the idea of decomposing the problem in smaller and easier subproblems. Relaxing the capacity constraints decomposes the problem in job-level subproblems. Relaxing the precedence constraints decomposes the problem in operation-level subproblems.

We now give an simple example that illustrates this approach.

Example
For simplicity, only one operation per order will be considered in this example. There are 2 machines, and 4 orders. Operation 1 of order $i = 1$ (1,1) is to be executed on machine $h = 1$. Operations (2,1), (3,1) and (4,1) of orders 2, 3, and 4 have to be executed on machine $h = 2$. The capacity of each machine type $M_{kh} = 1$ for all time periods. The due dates and weights of all orders and the duration of the (only) operation of these orders are given in this table:

Order	Due date	Weight	Duration
1	3	5	2
2	1	1	1
3	2	5	2
4	3	10	2

There are no precedence constraints, and thus no delays S_{ijl} either. The time horizon K is chosen big enough, e.g. $K = 8$.
The problem formulation is as follows

$$\begin{aligned} \text{minimise } J \;\; = \;\; & 5(\max(0, c_{11} - 3))^2 + (\max(0, c_{21} - 1))^2 + \\ & 5(\max(0, c_{31} - 2))^2 + 10(\max(0, c_{41} - 3))^2 \end{aligned}$$

subject to

$$\left(\sum_{i=1}^{4} \delta_{i1kh} \right) - 1 \;\leq\; 0 \qquad \text{for } k = 0, 1, \ldots, 8 \text{ and } h = 1, 2$$

$$\begin{aligned} c_{11} - b_{11} + 1 &= 2 \\ c_{21} - b_{21} + 1 &= 1 \\ c_{31} - b_{31} + 1 &= 2 \\ c_{41} - b_{41} + 1 &= 2 \;. \end{aligned}$$

To solve this problem, the capacity constraints are relaxed, and penalty terms are added to the objective function. Therefore, Lagrangian multipliers λ_{kh} are introduced, which can be

interpreted as the cost for machine h at time period k. This results in a modified problem formulation:

$$\begin{aligned}\text{minimise } J' \;=\; & 5(\max(0, c_{11}-3))^2 + \lambda_{b_{11},1} + \lambda_{b_{11}+1,1} + (\max(0, c_{21}-1))^2 + \\ & \lambda_{b_{21},2} + 5(\max(0, c_{31}-2))^2 + \lambda_{b_{31},2} + \lambda_{b_{31}+1,2} + \\ & 10(\max(0, c_{41}-3))^2 + \lambda_{b_{41},2} + \lambda_{b_{41}+1,2} - \sum_{k,h} \lambda_{kh}\end{aligned}$$

subject to

$$\begin{aligned} c_{11} - b_{11} + 1 &= 2 \\ c_{21} - b_{21} + 1 &= 1 \\ c_{31} - b_{31} + 1 &= 2 \\ c_{41} - b_{41} + 1 &= 2 \ . \end{aligned}$$

This problem is easier to solve, because it can be split up into 4 independent subproblems:

$$\begin{aligned}&\text{minimise } J_1 = 5(\max(0, c_{11}-3))^2 + \lambda_{b_{11},1} + \lambda_{b_{11}+1,1} \\ &\text{subject to } c_{11} - b_{11} + 1 = 2 \ ,\end{aligned} \tag{1}$$

$$\begin{aligned}&\text{minimise } J_2 = (\max(0, c_{21}-1))^2 + \lambda_{b_{21},2} \\ &\text{subject to } c_{21} - b_{21} + 1 = 1 \ ,\end{aligned} \tag{2}$$

$$\begin{aligned}&\text{minimise } J_3 = 5(\max(0, c_{31}-2))^2 + \lambda_{b_{31},2} + \lambda_{b_{31}+1,2} \\ &\text{subject to } c_{31} - b_{31} + 1 = 2 \ ,\end{aligned} \tag{3}$$

$$\begin{aligned}&\text{minimise } J_4 = 10(\max(0, c_{41}-3))^2 + \lambda_{b_{41},2} + \lambda_{b_{41}+1,2} \\ &\text{subject to } c_{41} - b_{41} + 1 = 2 \ .\end{aligned} \tag{4}$$

Given all λ values, each of these subproblems can be solved easily. If the initial values of the multipliers are all chosen equal to 1, then via complete enumeration the optimal value of b_{11} can be selected. For values of b_{11} ranging from 1 till 7, the corresponding values of J_1 are: 2, 2, 7, 22, 47, 82, 127. $b_{11} = 8$ is not a valid decision, since then c_{11} is larger than the time horizon K. Thus, an optimal value for b_{11} is 1. Similarly, also b_{21}, b_{31} and b_{41} are chosen to be 1, and this results in $J' = J_1 + J_2 + J_3 + J_4 - 16 = 2 + 1 + 2 + 2 - 16 = -9$. The resulting schedule is of course infeasible. Therefore, better values for the multipliers are required, such that J' is higher and closer to the optimal cost J of the original problem. A better set of multipliers is found by taking the derivative of J' to the λ values (the gradient, or — if the derivative does not exist — the subgradient $g(\lambda)$, where λ is the vector containing all λ_{kh} values). We have

$$g_{kh}(\lambda) = \left(\sum_{i=1}^{4} \delta_{i1kh} \right) - 1 \ .$$

For this example:

$$g(\lambda)^T = \begin{bmatrix} 1-1 & 1-1 & 0-1 & 0-1 & 0-1 & 0-1 & 0-1 & 0-1 \\ 3-1 & 2-1 & 0-1 & 0-1 & 0-1 & 0-1 & 0-1 & 0-1 \end{bmatrix} \ .$$

To find better multiplier values, λ is changed in the direction of the (sub)gradient, with a certain step size, e.g. $\alpha = 0.2$. How the step size is to be chosen can be found in [21]. The

new λ can be calculated as follows:

$$\begin{aligned}
\lambda^T &= \lambda^T_{\text{old}} + \alpha g(\lambda)^T \\
&= \begin{bmatrix} 1+0 & 1+0 & 1-0.2 & 1-0.2 & 1-0.2 & 1-0.2 & 1-0.2 & 1-0.2 \\ 1+0.4 & 1+0.2 & 1-0.2 & 1-0.2 & 1-0.2 & 1-0.2 & 1-0.2 & 1-0.2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 1.4 & 1.2 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} .
\end{aligned}$$

Remark that the multipliers have increased for the time periods that the schedule was infeasible, while they have decreased for the time periods where the machine was under-utilised. With these new multipliers, the subproblems (1) – (4) can be solved again. For J_1 and for b_{11} ranging from 1 till 7, the costs are 2, 1.8, 6.6, 21.6, 46.6, 81.6, 126.6. As such, $b_{11} = 2$ and $J_1 = 1.8$. For J_2 and b_{21} ranging from 1 till 8, the costs are 1.4, 2.2, 4.8, 9.8, 16.8, 25.8, 36.8, 49.8. For J_3 and b_{31} ranging from 1 till 7, the costs are 2.6, 7, 21.6, 46.6, 81.6, 126.6, 181.6. For J_4 and b_{41} ranging from 1 till 7, the costs are 2.6, 2, 11.6, 41.6, 91.6, 161.6, 251.6. Therefore, $b_{21} = 1$, $b_{31} = 1$ and $b_{41} = 2$. $J' = 1.8 + 1.4 + 2.6 + 2 - 14.2 = -6.4$, which is larger than in the previous iteration. One can see that the penalty for infeasibilities forces the operations to seek less favourite execution times, and as such reduces the amount of infeasibilities. This will go on until the schedule is almost feasible. A heuristic will then remove all leftover infeasibilities and produce a near-optimal performance. □

Remarks

While the relaxation of the capacity constraints works extremely well, the relaxation of the precedence constraints gives problems concerning solution oscillation between very early and very late beginning times. How these problems are solved, as well as a more elaborate description of the solution methodology can be found in [21, 22, 23, 29, 33].

Another problem is the fact that Lagrangian relaxation still takes a lot of time to converge. Therefore research is being done to implement the Lagrangian relaxation algorithm in a parallel program. In this version, by negotiating about the resource prices, the orders and the machines co-operate to produce a schedule that is near-optimal according to global performance criteria [17, 18].

6 Evaluation

In this paper we have tried to present some of the problems that can be solved by the ongoing research on DEDS systems in the interactions between our respective research groups. It is clear that the control of complex man-made systems such as manufacturing systems with a very high flexibility, transportation systems, etc. will require the solution of many different subproblems. We have presented some examples of techniques usable for ensuring safety constraints of different types, as well as methods for using the remaining freedom of choice in well-designed, safe systems, to schedule operations so as to maximise income.

For the first category of problems we have emphasised the use of Petri nets, even though in the literature also many other mathematical models are commonly used. We believe that the advantage of Petri nets is that they give a mathematical model that very closely resembles the physical lay-out of the system under study. The major difficulty in applying the results on discrete event systems (the same difficulty encountered in applying most theoretical results in

systems theory to control design!) is the problem of obtaining a good model. For Petri nets this is directly possible from a description of all the constraints in the physical system. Of course a lot of work is needed in finding 'good' models for black-box systems (e.g. when some components are proprietary design of a supplier, so that the user does not have a complete knowledge of the system) where only a limited set of input-output observations is available. Another problem is the simplification of Petri net models when it turns out that the model is too detailed and leads to too high a computational burden. Although there exist methods for speeding up the algorithms, even the best polynomial time algorithms will eventually become too time consuming. Therefore, model simplification is undoubtedly one of the important focuses of future research.

After modelling, control actions prevent the system from entering dangerous or unwanted situations. Control can be exerted implicitly or explicitly. Implicit control is achieved by extending the Petri net model. This extended Petri net constrains the course of events so that only allowable event sequences remain. Explicitly, controlled Petri nets prevent transitions from firing if the transition would possibly cause unwanted situations. Petri net control of real-world systems is achieved by mapping sensor signals and actuator commands to Petri net transitions.

Once a good controller has been found, guaranteeing all the safety requirements for the system, one has to design a higher level controller optimising some goal function. Here we have presented two different methods — max algebra and Lagrangian relaxation. Here again it is important to choose a model that includes all the important constraints, but does not make the model overly large. Otherwise even these efficient methods will run out of computation time.

Furthermore, as already said in the introduction, one of the major trade-offs that has to be taken into account when deciding which model to use is modelling power versus decision power. If we consider for example Petri nets, one of the most powerful mathematical modelling frameworks for DEDS, then a complete analytic solution is usually not available. Furthermore, the execution time of some algorithms explodes exponentially as the size of the problem increases. There are even some problems in connection with Petri nets for which there do not exist generally applicable algorithms that solve the problem. On the other side, the max algebra allows to say a lot about the properties of the system but it can only be applied to the subclass of DEDS that can be described by a max-linear time-invariant system.

The above trade-off is the Achilles heel for the industrial relevance of DEDS research. The problems for which execution times explode exponentially are precisely those problems for which industry awaits solutions and vice versa. Exploration of concepts that are potential solutions to this dilemma is vital for the research domain. One sample approach is to seek closer co-operation/integration of the systems that enforce safety (forbidden state avoidance) with optimising systems (maximising system benefits). In such an approach, the latter present the former with problems that can be handled efficiently.

References

[1] F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat, *Synchronization and Linearity.* New York: John Wiley & Sons, 1992.

[2] Z. Banaszak and B. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 724–734, Dec. 1990.

[3] R.K. Boel, L. Ben-Naoum, and V. Van Breusegem, "On forbidden state problems for a class of controlled Petri nets." To appear in IEEE Transactions on Automatic Control.

[4] R.K. Boel, L. Ben-Naoum, and V. Van Breusegem, "On forbidden state problems for colored closed controlled state machines," in *Proceedings of the 12th IFAC World Congress*, vol. 4, Sidney, Australia, pp. 161–164, July 1993.

[5] L. Bongaerts, "Report on specialisation in the use of Lagrangian relaxation for scheduling of flexible assembly systems," Internal Report 94R46, PMA, K.U.Leuven, 1994.

[6] Y. Brave and D. Bonvin, "A minimally restrictive policy for deadlock avoidance in a class of FMS," Tech. rep. 1992.03, Institut d'Automatique, EPFL, Lausanne, Switzerland, 1992.

[7] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot, "A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 3, pp. 210–220, Mar. 1985.

[8] G. Cohen and J.P. Quadrat, eds., *Proceedings of the 11th International Conference on Analysis and Optimization of Systems*, Sophia-Antipolis, France, vol. 199 of *Lecture Notes in Control and Information Sciences*, Berlin, Germany, Springer-Verlag, June 1994.

[9] R.A. Cuninghame-Green, *Minimax Algebra*, vol. 166 of *Lecture Notes in Economics and Mathematical Systems*. Berlin, Germany: Springer-Verlag, 1979.

[10] R. David, "Grafcet: A powerful tool for specification of logic controllers," *IEEE Transactions on Control Systems Technology*, vol. 3, no. 3, pp. 253–268, Sept. 1995.

[11] R. David and H. Alla, *Petri nets and Grafcet: Tools for Modelling Discrete-Event Systems*. London: Prentice-Hall, 1992.

[12] B. De Schutter and B. De Moor, "A method to find all solutions of a system of multivariate polynomial equalities and inequalities in the max algebra," Tech. rep. 93-71, ESAT/SISTA, K.U.Leuven, Leuven, Belgium, Dec. 1993. Accepted for publication in Discrete Event Dynamic Systems: Theory and Applications.

[13] B. De Schutter and B. De Moor, "Minimal realization in the max algebra is an extended linear complementarity problem," *Systems & Control Letters*, vol. 25, no. 2, pp. 103–111, May 1995.

[14] A. Falcione and B. Krogh, "Design recovery for relay ladder logic," in *Proceedings of the IEEE Conference on Control Applications*, Dayton, OH, Sept. 1992.

[15] M. Fisher, "An application oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, March–April 1985.

[16] P. Freedman, "Time, Petri nets, and robotics," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 417–433, Aug. 1991.

[17] L. Gou, T. Hasegawa, P. Luh, S. Tamura, and J. Oblak, "Holonic planning and scheduling for a robotic assembly testbed," in *Proceedings of the Rensselaer's 4th International Conference on Computer Integrated Manufacturing and Automation Technology*, New York, Oct. 1994.

[18] T. Hasegawa, L. Gou, S. Tamura, P. Luh, and J. Oblak, "Holonic planning and scheduling architecture for manufacturing," in *Proceedings of the International Working Conference on Cooperating Knowledge Bases Systems*, June 1994.

[19] Y.C. Ho, ed., *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*. Piscataway, New Jersey: IEEE Press, 1992.

[20] Y.C. Ho and X.R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*. Boston: Kluwer Academic Publishers, 1991.

[21] D. Hoitomt and P. Luh, "Scheduling the dynamic job shop," in *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, pp. 71–76, May 1993.

[22] D. Hoitomt, P. Luh, and K. Pattipati, "A Lagrangian relaxation approach to job shop scheduling problems," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 1944–1949, May 1990.

[23] D. Hoitomt, P. Luh, and K. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, pp. 1–13, Feb. 1993.

[24] L.E. Holloway and B.H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets systems," *IEEE Transactions on Automatic Control*, vol. AC-35, no. 5, pp. 514–523, May 1990.

[25] K. Jensen, *Coloured Petri nets: Basic concepts, analysis methods and practical use*. Berlin, Germany: Springer-Verlag, 1992.

[26] C. Jörns and L. Litz, "Hybrid modelling with Petri nets for the verification of logic control algorithms," in *Proceedings of the 3rd European Control Conference*, Rome, Italy, pp. 2035–2040, Sept. 1995.

[27] P. Kozák, S. Balemi, and R. Smedinga, eds., *Discrete Event Systems: Modeling and Control*, Progress in Systems and Control Theory, Birkhäuser Verlag, Basel, Switzerland, 1993. (Proceedings of the Joint Workshop on Discrete Event Systems (WODES'92), Aug. 26–28, 1992, Prague, Czechoslovakia).

[28] B.H. Krogh and L.E. Holloway, "Synthesis of feedback control logic for discrete manufacturing systems," *Automatica*, vol. 27, no. 4, pp. 641–651, 1991.

[29] P. Luh and D. Hoitomt, "Scheduling of manufacturing systems using the Lagrangian relaxation technique," *IEEE Transactions on Automatic Control*, vol. 38, no. 7, pp. 1066–1079, July 1993.

[30] G.J. Olsder, "Descriptor systems in the max-min algebra," in *Proceedings of the 1st European Control Conference*, Grenoble, France, pp. 1825–1830, July 1991.

[31] G.J. Olsder, "Eigenvalues of dynamic max-min systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 1, no. 2, pp. 177–207, Sept. 1991.

[32] G.J. Olsder, J.A.C. Resing, R.E. de Vries, M.S. Keane, and G. Hooghiemstra, "Discrete event systems with stochastic processing times," *IEEE Transactions on Automatic Control*, vol. AC-35, no. 3, pp. 299–302, Mar. 1990.

[33] T. Owens and P. Luh, "A time resolution improvement method for Lagrangian relaxation based scheduling algorithms," in *Proceedings of the 1993 NSF Design & Manufacturing Systems Grantees Conference*, Charlotte, North Carolina, Jan. 1993.

[34] Y. Peng and H. Van Brussel, "Modelling, simulation, and control of FMS using Petri nets," in *Proceedings of the Rensselaer's 4th International Conference on Computer Integrated Manufacturing and Automation Technology*, New York, pp. 248–253, Oct. 1994.

[35] J.L. Peterson, *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

[36] *Proceedings of the IEEE*, vol. 77, no. 1, Jan. 1989. Special issue on the dynamics of discrete event systems.

[37] K. Ravi and O.V. Krishnaiah Chetty, "Priority nets for scheduling flexible manufacturing systems," *Journal of Manufacturing Systems*, vol. 12, no. 4, pp. 326–340, 1993.

[38] P. Valckenaers, H. Van Brussel, and F. Bonneville, "Programming, scheduling, and control of flexible assembly systems," *Journal A*, vol. 34, no. 3, pp. 5–17, Nov. 1993.

[39] H. Van Brussel, Y. Peng, and P. Valckenaers, "Modelling flexible manufacturing systems based on Petri nets," *Annals of CIRP*, vol. 42, no. 1, pp. 479–484, 1993.

[40] M.C. Zhou and F. DiCesare, "Parallel and sequential mutual exclusions for Petri net modeling of manufacturing systems with shared resources," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 515–527, Aug. 1991.

[41] M.C. Zhou, F. DiCesare, and A.A. Desrochers, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 350–361, June 1992.

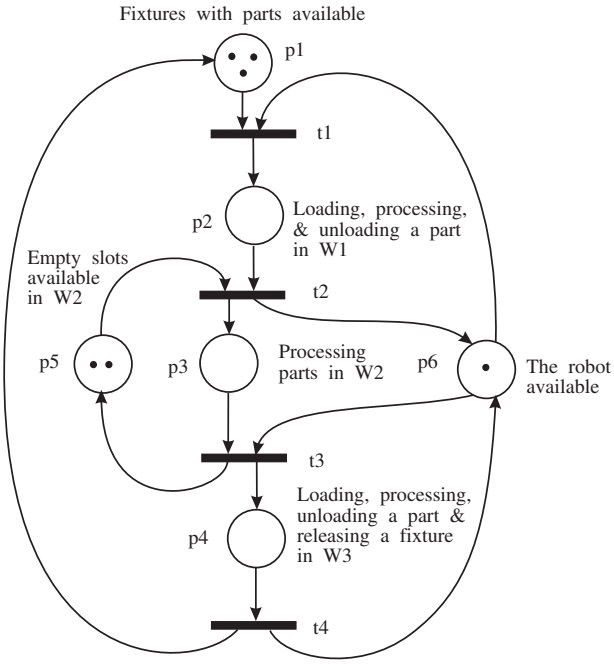

Figure 1: Petri net model of an assembly plant.

Figure 2: The PMA flexible assembly system testbed.

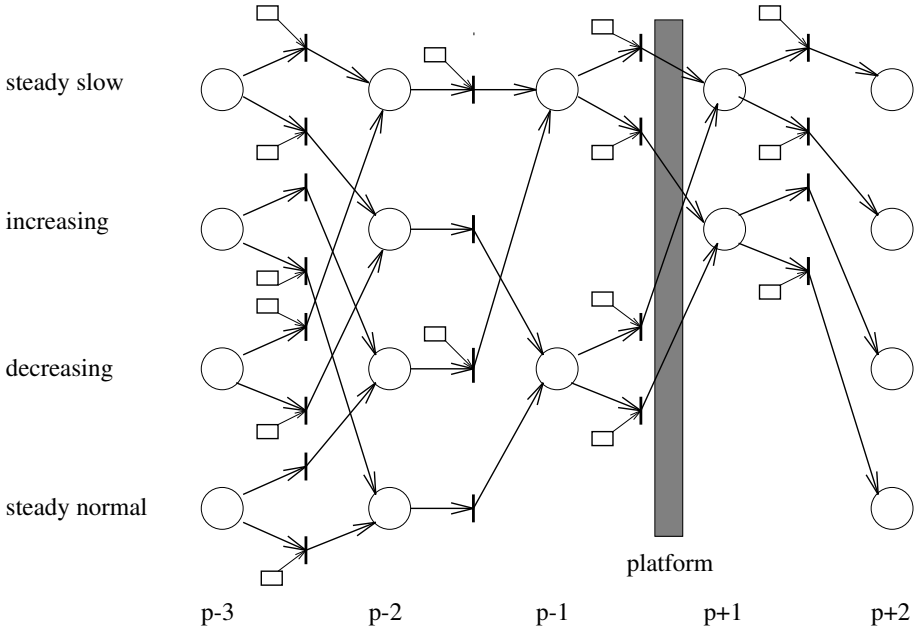

Figure 3: Metro line model.

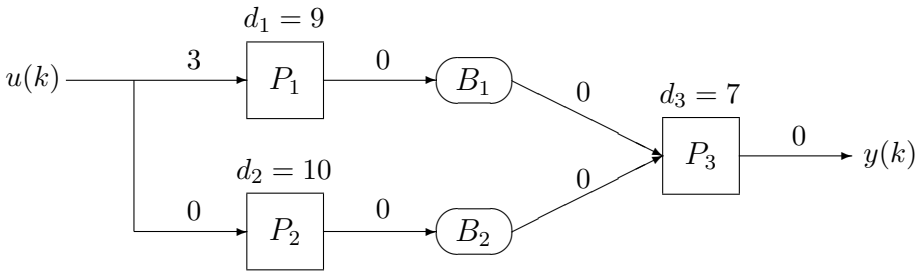

Figure 4: A simple production system.