UNIVERSITA' DEGLI
STUDI DI SALERNO

Unione Europea

Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione e Ingegneria Elettrica

Dottorato di Ricerca in Ingegneria dell'Informazione
VIII Ciclo – Nuova Serie

TESI DI DOTTORATO

Simulation-Based Control of Complex Material Handling Systems

CANDIDATO: **DOMENICO DEL GROSSO**

COORDINATORE: **PROF. MAURIZIO LONGO**

TUTOR: **PROF. PASQUALE CHIACCHIO**
COTUTOR: **ING. FRANCESCO BASILE**

Anno Accademico 2008 – 2009

Copyright © by Domenico Del Grosso 2010
All Rights Reserved

Contents

List of Figures

Acknowledgements

My first thanks go to my thesis supervisors Pasquale Chiacchio and Francesco Basile for their being a fundamental guide to my research along the last four years. Their presence and respect, joint with professionalism and patience, have enriched my work day by day.

I thank my family, Roberta and my acquired family for their unconditional love. Without you all, I'd never had reached this and many other objectives in my life. Word cannot say how much you are important to me.

At last I thank all my dearest friends and colleagues for making more pleasant the game of life. I know I would not forget any of you, but I don't want to make lists. You know who you are...

To my family for always being there
Supporting me and giving me strength
Day by day, even in the worst moments
And to who I find every time I raise my sight..

Chapter 1

Introduction

Material Handling (MH) consists in the movement and storage of parts, in a manufacturing or distribution process, from one location to another.

Figure 1.1 Manned Material Handling Systems: Forklift Truck.

Material Handling Systems (MHSs) are everywhere in production plants, assembly lines, product distribution, logistics, intermodal activities (railways, road transportation, container ships, etc..). They usually are distributed, sometimes itinerant and often mixed manned and automated.

Figure 1.2 Automated Material Handling System.

It has been estimated that MH accounts for up to 80 percent of production activities ([App77], [HE09]). Although not adding value in the manufacturing process, MH usually influences great part of a company's operation costs, especially, for example, in the food distribution chain. Due to the increasing demand for a high variety of products, *flexibility* and *efficiency* are two important keywords in MHSs.

Optimizing MH activities means having shorter response times and an increased throughput of the plant. The importance of this optimization process is very high in today's companies. Nowadays, the interest in this process is growing rapidly since several new technologies, like the Radio Frequency Identification (RFID) are

available which finally allow to introduce an automation level to operating MHSs, almost without stopping operations and at a very low cost.

Figure 1.3 Itinerant Material Handling System.

In MHSs *planning* consists in managing decisions that affect the middle-term activities (one or multiple months), such as inventory management and storage location assignment [Van99], or the design of the warehouse system itself. *Control* problems, instead, involve the problem of optimal sequencing and scheduling of short-term activities, which will be ahead called "problem of dispatching".

For control purposes, a *model* of the system is necessary. Due to the complex and heterogeneous nature of MHSs, modeling ap-

proaches proposed in the literature are typically very specific and context-dependent. Moreover, the strong combinatorial nature of the control problem, and the presence of a great number of constraints to be considered, usually make the design of a control solution very tough. To devise a closed form analytical control action can require a great computational effort and could result not so convenient ([GZNL08]). Indeed, turbulence and variations in the input set of the system can suddenly make not more adequate a hardly designed control action.

Thus, the choice of *Dispatching rules* as control actions, despite producing only local optimum solutions, is very usual for MHSs. Dispatching rules, indeed, result in a more reasonable and robust way to control MHSs since they are effective and computationally inexpensive.

In the absence of a closed form control solution, *Simulation* is fundamental to evaluate the effects of a control action which cannot be analytically predicted. The outcome of the application of a rule or another can be easily tested via simulation and this is the reason why having a good model assumes a further major importance.

In this thesis the problems of the modeling and of the control of complex MHSs are faced and an effective and efficient modeling and control architecture is proposed. The problem statement and the main contribution of this work, w.r.t. the existent literature on MHSs, will be more clear in the following paragraphs of this chapter.

1.1 Problem Statement: Material Handling Systems

In a MHS there are several parts, or items, to be moved and resources that can execute movement tasks, also called *Missions*. The basic problem in MHSs consists in determining which mission must be executed by available resources, in order to obtain the best mission-to-resource assignments over a certain time horizon.

In the systems considered in this work the parts to be handled can be:

1. *Unit Loads* (UL): parts that cannot share the resource with other parts in the same travel;

2. *Less Than Unit Loads ([Mal98])* (LTUL): parts that can be handled contemporarily by a resource, since they fill only a portion of the resource.

Missions are supposed to be contained into a *Buffer* which can be refilled every time a new batch of missions becomes available. Typically, the arrival of missions in a MHS is totally asynchronous and random.

Resources, can be:

1. *Automated Vehicles* (Automated Guided Vehicles, Rail Guided Vehicles): Resources that are completely automated and that don't require the presence of the human operator;

2. *Manned Vehicles*:
   - Motor Driven Vehicles (Motor Pallet Trucks, Counterbalanced Fork-lift Trucks): Resources that can be ridden by an operator and that are used for long distances and high weights;
   - Manually Operated Vehicles (Handcarts, Hand Trucks and Hand Pallet Jacks): Resources that are used by storemen for shorter transportation distances;

3. *On-foot Storemen*: Men that pick small objects one by one and that collect them typically in bags.

Between resources there can be two kinds of differences:

- Differences that make them belong to disjointed compatibility groups (Fig. 1.4a): not every resource, indeed, is necessarily able to execute every mission (for example a fork-lift truck cannot handle small objects).

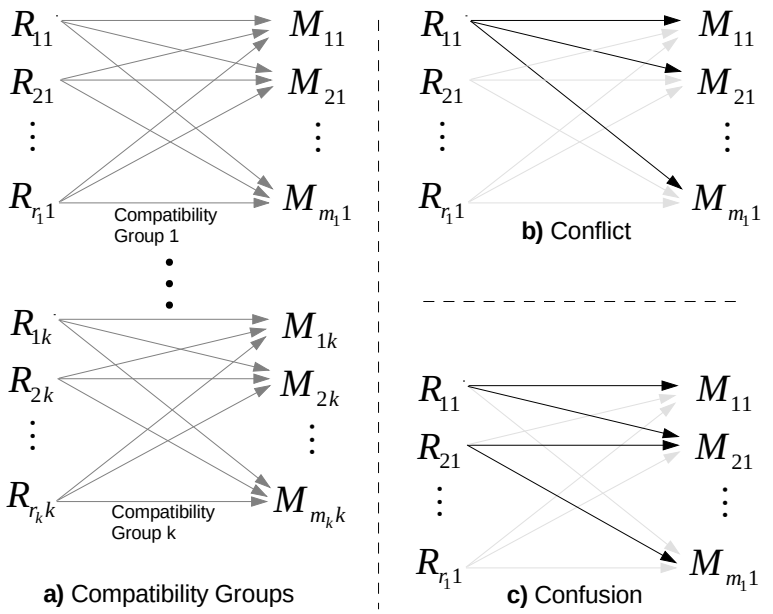

Figure 1.4 Mission-to-Resource associations in MHSs.

- Differences that make them behave differently even if belonging to the same compatibility group, i.e. resources performances like speeds, capacities, etc...

When one resource is available for a set of missions (Fig. 1.4b) a *Conflict* occurs. When two or more resources are simultaneously available (Fig. 1.4c), several conflicts can be mutually dependent, in the sense that the solution of a conflict influences the others. In this case a *Confusion* occurs.

The main control problem in MHSs consists in determining *Who* has to do *What* and *When* in a manner such that a certain objective is reached over a certain time horizon.

1.2 Discussion on the main terms

In order to better understand some topics discussed in the reminder of this work, the main terms, involved in the discussion,

are now presented.

- *Dispatching*: the so-called problem of "Dispatching" consists in defining a procedure to assign resources to missions. This is often made by using heuristic rules called *Dispatching rules*.

- *Real-Time Control*: it consists in having a control action that can be adapted in real-time to the variations of the input data, of the system's characteristics and to the presence of disturbances.

- *Simulation-Based Control*: simulation-based control approaches consist in using simulation to evaluate a control strategy before it is applied to the real system. This is a quite common control technique in complex manufacturing systems.

- *Event-Driven and Periodic Scheduling*: event-driven and periodic scheduling are the two main reactive scheduling strategies for the real-time control of MHSs. Event-driven scheduling consists in rescheduling the activities when an event, like the unforseen arrival of a new batch of missions, occurs. Periodic scheduling consists, instead, in rescheduling the activities on fixed time instant. The discussion on which one outperforms the other is still open, but event-driven scheduling approaches are more often preferred to periodic ones.

- *Reconfigurable Material Handling Systems*: the reconfiguration of a system can be:

  - "soft" when the tuning of some system's parameters is used to optimize its performances;
  - "hard" when even the physical layout of the system is modified to improve the performances.

  Reconfigurable systems have gained much interest in the last years. This is due to:

  - the increasing computational power of computers which allows to develop flexible software models and controllers;
  - the increasing use of flexible machines and plant devices in modern facilities.

- *Automated Model Generation*: it consists in the possibility to automatically generate, through a program or an algorithm, the model of a system starting from the minimal set of necessary information. It is particularly useful for reconfiguration purposes, when a system's model must be recomputed.

- *Operational Control*: operational control includes decisional activities like the task sequencing for resources.

- *Supervisory Control*: supervisory control problems typically involve issues like mutual exclusion in sharing resources and deadlock resolution. In this work it is assumed that deadlocks cannot occur since resources are machines locally controlled by deadlock avoidance policies, or human operators which naturally do not incur deadlocks. Thus, supervisory control issues will not be considered.

1.3 Literature Survey

Good introductions to MHSs can be found in [Vis02], [CT09], [HE09] and [Rev05]. MHSs concepts are directly derived from the Flexible Manufacturing Systems (FMSs) control theory which has been widely treated in the literature.

In [HR91] and [Har95] a review of the 80's and half 90's research on the issue of the real-time scheduling in the computer based control is presented. Among the scheduling techniques proposed in the literature of MHSs, dispatching rules have been widely proposed. They are a frequent control solution for the real-time control of manufacturing systems in general ([Vis02], [Pin05], [Van00], [Pro07], [JR98], [KK94], [JK98], [HL06], [JM98]).

The idea of real-time control in flexible manufacturing systems has been firstly introduced in practice by Yamamoto and Nof. In [YN85] Yamamoto and Nof face the problem of the variability in the scheduling process of dynamic manufacturing systems from a practical point of view. The authors develop a real application built by using a mainframe computer to obtain, at real-time, a nearly-optimal schedule of part-mix schedules on several machines, which can breakdown. The actual progress data of the operations is continuously compared with the current schedule and, if the difference exceeds a specified limit, or if a machine is in trouble, the schedule is revised. Thus, reschedules are not planned in advance. Three rescheduling techniques are considered:

1. Follow strictly the sequence of the original schedule;
2. Apply priority dispatching rules;
3. Complete reschedule the activities.

In the latter case, a schedule tree, which includes all the scheduling combinations, is produced by the mainframe. The objective function defined for the research of the optimum solution along the tree, is the minimization of the total processing time. The computer is used to find the optimal solution. This solution is a nearly-optimal solution since authors use approximated research methods to find the optimum solution along the tree.

Wu and Wysk present in [WW89] a simulation-based mechanism for the online control and the dynamic scheduling of a flexible manufacturing system. In their work, the authors draw the attention on the fact that a combination of simple dispatching rules outperforms a fixed dispatching rule. For instance, they demonstrate that a fixed dispatching rule which minimizes time, optimizes a single index like the execution time but, at the same time, it usually produces late jobs. Thus, they suggest to always combine different dispatching rules together to improve the general performance of the system. The authors develop an architecture where discrete event simulation is used to test several dispatching rules over a short-term period in order to dynamically choose

the one the optimizes a certain index. Consequently a continual alternation of different dispatching rules is carried out automatically. Simulations are based on a physical description of the system which is obtained from separate data files that include part data, machine data, system data, etc.. Wu and Wysk notice that the decision made in a short-term period is local and that it may not contribute positively to the global system performance. Nevertheless, the suggest that in a highly dynamic system like a FMS, the system status changes over time and a static and globally optimum schedule is too hard to be produced.

Not too differently by [WW89], Kim et al. in [KK94] propose a simulation model, written in FORTRAN language, which allows to dynamically vary the dispatching rule. The evaluation of the effects of candidate rules is done by using simulation. In addition to [WW89] Kim assumes that a performance measure of the system is given and that "minor" and "major disturbances" can occur which alter this measure thus inducing a rescheduling need. The selection of the rule is done at the beginning of each planning horizon or in presence of major disturbances. When the so-called "rule selector" calls the simulation mechanism, several discrete event simulations are performed with each rule and the rule selector selects the best one for a given performance measure. The rule selected by the rule selector goes to the scheduling controller as an input of the control system. Thus, simulation is used as a decisional support. Kim also provides a detailed inspection of simulation results with several performance measures.

Jeong in [JK98] proposes a more systematic framework than [KK94] for the real-time scheduling in manufacturing systems using simulation and dispatching rules.

In [SRW99] the idea of [WW89] is realized using a more modern simulation tool which is the Arena© software. The authors firstly present a review of the literature on real-time scheduling and on simulation for shop floor control, stressing the importance of having a combination of rules over a system's production cycle in order to have better performances. They define the "Multi-Pass Simulation Based Scheduling" as the use of discrete event simula-

tion to evaluate a set of feasible control policies for a short planning horizon, and determining the one that impacts the system most favorably. The authors highlight that, to use simulation as a look-ahead for real-time control decisions, it must be fast enough to not disrupt the operation of the system. Thus, they call it "fast simulation mode". Eventually, they propose a control framework for shop floors in which, when the look-ahead simulation is activated, a number of alternative models are run and evaluated. However, while the simulation model is periodically or event-based changed, the dispatching rule is fixed once at the startup of the system and never varied. The model, being not a formal model, is validated by comparing the simulation results with the performance measures of the physical system.

Son and Wysk in [SWJ03] enhance the approach proposed in [SRW99] by defining and developing a more formal simulation model for Arena©. The model is built through an automated procedure which combines together some information like the so-called "Default Components" information, "Static information", "Dynamic information", etc.. Eventually, the authors propose to use the simulation model as a supervisor interacting with the physical equipment through an "executor". Then, they suggest to use the model for a simulation-based control. In order to validate the methodology they test it on six manufacturing systems, providing also some benchmark results.

In the ambit of the real-time control of manufacturing systems the difference between periodic and event-driven rescheduling policies has been well addressed by Church and Uzsoy in [CU92]. The authors compared these two scheduling techniques and demonstrated, through analytical results, that event-driven policies produce better schedules than periodic rescheduling policies.

Naso et al. in [NST07] focus on the problem of the reactive scheduling in a distributed network for the supply of perishable products. The authors propose an event-driven rescheduling technique based on the observation of major perturbations in the process. Moreover, in [NST07] a detailed mathematical model of MHS, based on algebraic expressions, is proposed which, however,

is too tailored on a specific case study.

An important topic in the real-time control theory are reconfigurable systems. The reconfiguration of a system can consist in simply tuning some process or control parameters, or it can involve the physical system when flexible hardware is available. Typically, flexible layouts or tunable manufacturing environments allow a system to be reconfigured in order to be dynamically adapted to the control requirements. Li et al. in [LDM09] propose a supervisory controller for manufacturing systems, which can be easily reconfigured in order to react to the perturbations. MHSs, in particular, are treated by Jokinen in [JM07] and the use of flexible and variable layouts in MHSs for reconfiguration purposes is studied by Wong et al. in [WTZ+07]. Two situations are usually considered for the reconfiguration of a MHS:

1. On machines breakdowns, i.e. as a resource fails, the physical system cannot stop working and it has to be reconfigured. Consequently the work has to be rescheduled, based on a different system's structure;

2. Given the high variability of the input jobs list, when an expected job is late (a truck's arrival has been delayed) or when there is an unexpected urgent jobs arrival, the current schedule does not match anymore with the actual availability of jobs and the work has to be rescheduled.

The importance of formal methods for the modeling of manufacturing systems and the controller synthesis is a well addressed topic in the literature ([CVP99], [Kow99], [BCD08b], [BCD08a], [BCD09c]). Formal modeling and control methods have been widely proposed for Automated Warehouses ([PDL99], [Cra97], [LS96]). Several authors used Petri Nets (PNs) and Colored Petri Nets (CPNs) as formal modeling tool ([ABCC05], [DF04], [ZFY05], [KD91], [RVL92], [VRL94]).

Having an automated model generation methodology is very important for the real-time control of manufacturing systems, especially for reconfiguration purposes. An example of formal procedure to automatically generate the model of a shop floor system

is proposed by Son and Wysk in [SWJ03] but, in general, few contributions exist in the literature of MHSs.

Among formal modeling methodologies, an appealing approach is the matrix-based framework proposed by Giordano, Lewis, Naso et al. Lewis in [TL97] proposes a matrix model for the simulation of Discrete Event Systems in general. A more control-oriented approach is presented in [ML01] and [BLK+02]. Applications to MHSs can be found in [GZN+06b], [GZN+06a] and [GZNL08] where a variable dispatching rule control approach is used for operational control issues. Matrices are used to obtain a formal model of the entire system by assembling the discrete event models of the atomic components. The authors also show that the matrix-based model can be included into a multi-level control architecture in which the model is used to determine when a control decision has to be made from upper levels, and also to feature operational control tasks. Simulation is used to tune, off-line, some parameters of the control law. Then, this fixed weighted combination is adopted as control strategy. The relationship between matrix models and PNs are discussed in [PJ05] and [BLKM06].

1.4 Contribution of the Work

At first let us make some general considerations on the past literature:

- Both the issue of the modeling and of the control of MHSs are not new in the scientific literature. In particular, the simulation-based real-time control of MHSs is a very hot topic.

- Several modeling methodologies have been proposed for MHSs. For example, PNs are widely used as formal modeling tool in the scientific literature of manufacturing systems. However, PN-based modeling approaches mainly involve automated warehouses and lack in generality being too layout-dependent. In general, modeling approaches for MHSs are

mainly focused on automated warehousing systems, or they are not too suited to simulation and to conflict detection or they propose models which are too tailored to specific contextes. However, it is hard to find literary contributions in which the advantages of formal models are joint with simulation-based control techniques for the real-time control of MHSs.

- An appealing and novel modeling and control approach for MHSs has been devised by Giordano et al. in [GZNL08]. In the approach of Giordano et al. dispatching decisions are made on-line with the system using real-time information to identify the most appropriate control action at any time, thus facing the turbulence of the environment. Via simulation of the system the parameters of a control objective function are off-line tuned. The main limit of this approach is that decisions are local and the best control action is chosen without looking ahead the effects over time.

- Good simulation-based control methodologies are proposed in [KK94], [JK98] and [SWJ03]. Their main limit is that they do not formally approach the modeling of the systems to be controlled, and simulations usually exploit commercial simulation tools like Arena©. As a consequence, these approaches are not enough general to be applied to every MHSs.

Based on the results presented in [BCD09b] and [BCD09a], the main contribution of this thesis is to propose a formal and general modeling methodology for complex MHSs which is merged with a simulation-based control framework into a unique modeling and control architecture.

The proposed formal model is based on Colored Timed Petri nets (CTPNs) and

- it is used to model the activities in the system with a structure that is the same for every MHS and to detect conflicts and confusions;

- it is extended with an additional modeling level (see Fig. 1.5), called *Auxiliary Process* (AP) which captures the high level information, and the constraints on the execution of the missions, which are specific for each plant;
- it allows to easily express, and implement, dispatching rules;
- it is low computational demanding thus allowing to be used for on-line control purposes;
- it requires low customization efforts, thus reducing modeling costs, since it is very general and flexible;
- it permits to formally characterize the system's state. This makes possible to realize halt and recover mechanisms in which the state of the system is saved and restored when necessary;
- it can be automatically built and re-built whenever something in the system changes, thanks to an automated model generation procedure (this is useful for reconfiguration and rescheduling purposes).

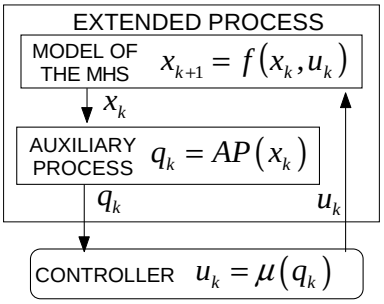

Figure 1.5 Extended Process: Formal model of the MHS and Auxiliary Process.

Simulation, instead, is used

- to always know who is involved in control decisions and when such decisions must be made;
- to test in advance several dispatching rules over a certain time horizon, in a look-ahead fashion, in order to dynamically vary the control action over time by always choosing the best dispatching rule, at a very low cost;
- to implicitly overcome the problem of the feasibility of control solutions highlighted in [Smi95], since unfeasible control directions are not included.

The term "complex" is here used to denote MHSs which are very turbulent and heterogeneous, i.e. MHS in which the mission set and the resource set are variable, and items and resources are strongly heterogeneous. The consequent highly combinatorial and hardly constrained nature of the control problem makes simulation the only way to predict the effects of a control policy.

To simulate the CTPN model of the system, the PNs simulation tool PNetLab ([BCC07]) is used. The developed control architecture is validated through several simulations based on real case studies.

The proposed approach is ready to be used for the online control of real MHSs in which measures from the plant are available (to this purpose RFID technology, for example, can come in aid [BCDD06]). In such an online control framework, thanks to Petri nets, also fault diagnosis issues could be considered ([BCDD08]).

Low level control problems are neglected since the attention is focused on the general problem of the assignment of jobs to resources. Moreover, in the proposed architecture it is assumed that deadlocks cannot occur since resources are machines locally controlled by deadlock avoidance policies, or human operators which naturally do not incur deadlocks. Thus, supervisory control issues will not be considered.

1.5 Overview of the Thesis

The thesis is organized as follows:

- In Chapter 2 the theoretical background of this thesis will be provided. First of all, PNs and CPNs fundamentals, necessary for the comprehension of the work, will be presented. Then, the concept of Scheduled Event List, which is a key element of the simulation of Discrete Event Systems, will be briefly presented.

- In Chapter 3 the modeling framework for MHSs proposed in this work will be described. The description will go through the two main components of the modeling framework, i.e. the *formal model* and the *auxiliary process*. Some examples will be used to clarify the modeling approach.

- In Chapter 4 the proposed *model-based control environment* will be firstly described. Then, the algorithm devised for the detection of confusions in a CPN will be presented and clarified through an example. A description of the objective functions involved in the optimization process will precede a review of the most common dispatching rules for MHSs, whose formal expression will be provided. Then, an *operational control scheme*, based on the model-based control environment, will be presented. Finally, on-line control issues for MHSs will be discussed. In particular the possibility to have reschedules of the activities, based on a reconfigured model when some perturbations occur, will be considered.

- In Chapter 5 several simulations will be presented to validate and to show the effectiveness of the proposed approach. At first two simulation strategies (called batch simulation and single buffer simulation) will be compared. Then a comparison among the dispatching rules will be performed. Finally, the operational control scheme previously presented will be fully exploited by fixing an objective function and letting the dispatching rule vary along a series of batches of missions.

Also the online control in case of disturbances will be tested and discussed.

- In Chapter 6 the main conclusions of the work will summarized together with a discussion on the possible future research issues.

Chapter 2

Fundamentals

In this chapter the theoretical background of the thesis is provided. PNs are firstly introduced. Then, a brief overview on CPNs is furnished. However, for further details on PNs and on simulation of PNs, the reader can refer to [Mur89] and to [BCC07]. At last, an overview on the simulation of Discrete Event Systems [CL08] is furnished.

2.1 Background on Petri Nets

An ordinary *Place/Transition* net (P/T net) is a 4-tuple

$$N \; = \; (P, T, \mathbf{Pre}, \mathbf{Post}).$$

- P is a set of m places (represented by circles);
- T is a set of n transitions (represented by bars);
- $\mathbf{Pre} : P \times T \mapsto \mathbb{N}$ ($\mathbf{Post} : P \times T \mapsto \mathbb{N}$) is the *pre* (*post-*) *incidence* matrix;
- $\boldsymbol{C} \; = \; \mathbf{Post} - \mathbf{Pre}$ is the incidence matrix;
- The net *marking* is represented as a vector $\boldsymbol{m} \in \mathbb{N}^m$. The marking of a place is a scalar value $m_i \in \mathbb{N}$. A transition t is enabled at $\boldsymbol{m}$ iff $\boldsymbol{m} \geq \mathbf{Pre}(\cdot, t)$ and this is denoted as

$\boldsymbol{m}[t\rangle$. An enabled transition t may fire yielding the marking $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}(\cdot, t)$ and this is denoted as $\boldsymbol{m}[t\rangle\boldsymbol{m}'$;

- The symbols ${}^\bullet p$ (${}^\bullet t$) and $p^\bullet$ ($t^\bullet$) are used for the *pre-set* and *post-set* of a place $p \in P$ (transition $t \in T$), respectively, e.g.

$$ {}^\bullet t = \left\{ p \in P \mid \mathbf{Pre}(p, t) \neq 0 \right\};$$

- An ordinary net N is a Marked Graph (MG) if ${}^\bullet p = p^\bullet = 1$, $\forall p \in P$;

- An ordinary net N is a Free Choice Net (FCN) if $\forall p \in P$ $|p^\bullet| \leq 1$ or ${}^\bullet\{p^\bullet\} = p$;

- A P/T system $\langle N, \mathbf{m_0} \rangle$ is a P/T net N with an initial marking $\mathbf{m_0}$;

- A *firing sequence* from $\mathbf{m_0}$ is a (possibly empty) sequence of transitions $\sigma = t_1 ... t_k$ such that $\mathbf{m_0}[\mathbf{t_1} > \mathbf{m_1}[\mathbf{t_2} > \mathbf{m_2} ... [\mathbf{t_k} > \mathbf{m_k}$;

- A marking $\mathbf{m}$ is reachable in $\langle N, \mathbf{m_0} \rangle$ iff there exists a firing sequence σ such that $\mathbf{m_0}[\sigma > \mathbf{m}$. Given a net system $\langle N, \mathbf{m_0} \rangle$ the set of reachable markings is denoted $R(N, \mathbf{m_0})$;

- The function $\sigma : \mathbf{T} \to \mathbb{N}$, where $\sigma(\mathbf{t})$ represents the number of occurrences of t in σ, is called firing count vector of the fireable sequence σ. If $\mathbf{m_0}[\sigma > \mathbf{m}$, then it is possible to write in vector form

$$ \boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}(\cdot, t) \cdot \sigma $$

, known as *state equation* of the system;

- A P/T system is *live* when, from every reachable marking, every transition can occur. N is *structurally live* iff $\exists \mathbf{m_0}$ such that $\langle N, \mathbf{m_0} \rangle$ is live;

- A place $p \in P$ is said to be *k-bounded* iff $\forall \mathbf{m} \in \mathbf{R}(\mathbf{N}, \mathbf{m_0})$, $\mathbf{m}(\mathbf{p}) \leq \mathbf{k}$. A net system $\langle N, \mathbf{m_0} \rangle$ is said to be *k-bounded* iff

each one of its places is k-bounded, and it is bounded iff it is bounded for some $k \in \mathbb{N}$. A net N is *structurally bounded* iff $\forall \mathbf{m_0}$ the net system $\langle N, \mathbf{m_0} \rangle$ is bounded. N is *structurally bounded* iff $\exists \mathbf{x} \in (\mathbb{N}^+)^{\mathbf{m}}$ such that $\mathbf{x^T C} \leq \mathbf{0}$;

- N is *conservative* iff $\exists \mathbf{x} \in (\mathbb{N}^+)^{\mathbf{m}}$ such that $\mathbf{x^T C} = \mathbf{0}$.

As shown in Fig. 2.1, a *structural* conflict exists when

$$^\bullet t_i \cap ^\bullet t_j \neq \emptyset.$$

If t_i and t_j are both enabled, the conflict is said *behavioral*.

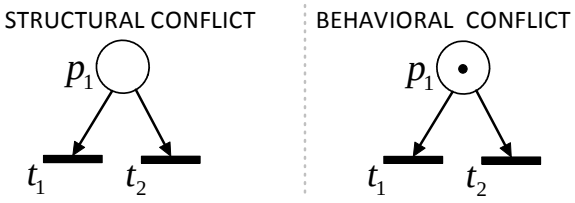

Figure 2.1 Conflict in Petri nets.

A confusion occurs when two or more groups of conflicts are mutually dependent as shown in Fig. 2.2. When there is a confusion, the way a conflict is solved influences also the existence itself of the other conflicts.

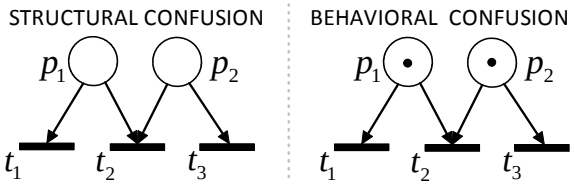

Figure 2.2 Confusion in Petri nets.

It is also possible that more than two groups of conflicts are mutually dependent, thus leading more confusions in the same PN which are in a bind relation among them (see Fig. 2.3).

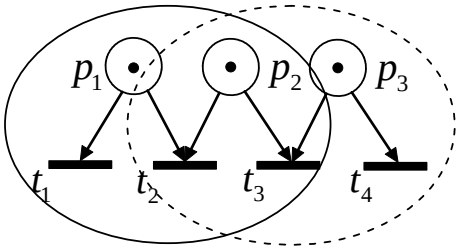

Figure 2.3 Couple of mutually dependent confusions in Petri nets.

2.2 Background on Colored Petri Nets

All the formal definitions given for PNs can be naturally extended to CPNs which are high level PNs. A *CPN* is a 6-tuple

$$N \;=\; (P, T, \mathbf{Pre}, \mathbf{Post}, Cl, Co).$$

Remark: In literature, more than one formal definition for CPNs exist, depending on how the incidence matrix and transition colors are defined. In the formalism chosen in this work, incidence matrix entries are represented by matrices.

- P is a set of m places (represented by circles);
- T is a set of n transitions (represented by bars);
- Cl is a set of colors;
- Co: $P \cup T \longrightarrow Cl$ is a color function that associates to each element in $P \cup T$ a non-empty ordered set of colors in the set of possible colors Cl. $\forall p \in P, Co(p_i) = \{a_{i,1}, a_{i,2}, ..., a_{i,u_i}\} \subseteq Cl$ is the ordered set of possible colors of tokens in p_i, and u_i is their number. $\forall t \in T, Co(t_j) = \left\{b_{j,1}, b_{j,2}, ..., b_{j,v_j}\right\} \subseteq Cl$ is the ordered set of possible occurrence colors in t_j, and v_j is their number;
- **Pre** and **Post** are the pre-incidence and post-incidence $m \times n$-sized matrices, respectively. $\mathbf{Pre}(p_i, t_j)$ is a mapping from the set of occurrence colors of t_j to a non-negative multiset

over the set of colors of p_i, namely, $\mathbf{Pre}(p_i, t_j) : Co(t_j) \rightarrow \mathbb{N}(Co(p_i))$, for $i = 1, ..., m$ and $j = 1, ..., n$. $\mathbf{Pre}(p_i, t_j)$ represents a matrix of $u_i \times v_j$ non-negative integers whose generic element $Pre(p_i, t_j)(h, k)$ is equal to the weight of the arc from place p_i w.r.t color $a_{i,h}$ to transition t_j w.r.t color $b_{j,k}$. $\mathbf{Post}(p_i, t_j) : Co(t_j) \rightarrow \mathbb{N}(Co(p_i))$, for $i = 1, ..., m$ and $j = 1, ..., n$. $\mathbf{Post}(p_i, t_j)$ represents a matrix of $u_i \times v_j$ non-negative integers whose generic element $Post(p_i, t_j)(h, k)$ is equal to the weight of the arc from transition t_j w.r.t color $b_{j,k}$ to place p_i w.r.t color $a_{i,h}$;

- The incidence matrix C is a $m \times n$ matrix, whose generic element $C(p_i, t_j) : Co(t_j) \rightarrow \mathcal{Z}(Co(p_i))$, for $i = 1, ..., m$ and $j = 1, ..., n$, is the $u_i \times v_j$ matrix of integer numbers $C(p_i, t_j) = \mathbf{Post}(p_i, t_j) - \mathbf{Pre}(p_i, t_j)$;

- For each place $p_i \in P$, the marking m_i is defined as a non-negative multi-set over $Co(P_i)$. The mapping $m_i : Co(P_i) \rightarrow \mathbb{N}$ associates to each possible token color in P_i a non-negative integer representing the number of tokens of that color that is contained in the place p_i. The column vector of u_i non-negative integers, whose hth component $m_i(h)$ is equal to the number of tokens of color $a_{i,h}$ that are contained in p_i, is denoted as $\mathbf{m}_i$. The marking of a CPN is an m-dimensional column vector of multisets:

$$\mathbf{m} = \begin{bmatrix} \mathbf{m}_1 \\ ... \\ \mathbf{m}_m \end{bmatrix}.$$

  For the sake of simplicity, a token of color "c1" contained in a place p_i will be indicated with the symbol $(c1)$;

- The concepts of *pre-set* and *post-set* of a place $p \in P$ or a transition $t \in T$ are naturally inherited from PNs, but colors must be also considered:

$$^{\bullet}t_i c_j = \{t_i c_j \in T \mid \mathbf{Pre}(p_h c_k, t_i c_j) \neq 0\}.$$

A confusion in CTPNs is shown in Fig. 2.4, where the transition t_2 under the color $mr1$ is in confusion with t_1 under the color $mr1$, t_{G1} under the color $mr1$ and t_2 under the color $mr2$. Instead t_2 under the color $mr2$ is in confusion with t_3 under the color $mr2$, t_{G2} under the color $mr2$ and t_2 under the color $mr1$.

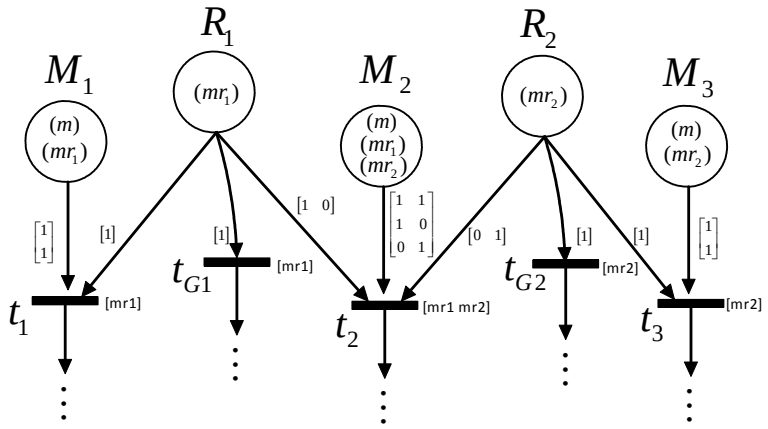

Figure 2.4 Confusion in Colored Petri nets.

When time is added to PNs and CPNs a time function is defined, which associates to each transition t_i in the case of PNs, or to each transition color $t_i c_j$ in the case of CTPNs, a time duration from enabling to firing. In this case the PNs and CPNs become TPNs and CTPNs. Notice that timed and un-timed transitions will be represented with empty filled boxes and black bars, respectively.

2.3 Simulation of Discrete Event Systems

Simulation is a key instrument for this work. The simulation of Discrete Event Systems (DESs) is based on the Scheduled Events List (SEL) [CL08] which is a list containing the set of enabled events at each time instant. Moreover, each control action is carried out by modifying the contents of the SEL. Thus, it is worth

to briefly describe the way the SEL works, but for further details the reader can refer to [CL08].

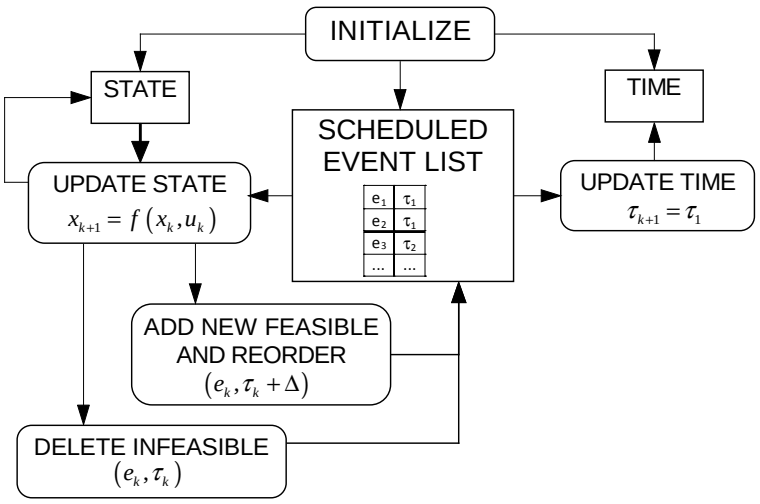

Figure 2.5 Scheduled Event List.

At each time instant, the SEL (see Fig. 2.5) contains all the enabled events. If these events are "timed", their firing delay τ is also held by the SEL, together with the corresponding event. As it will be shown in the remainder of the work, the events included into the SEL correspond to the missions for which an available resource exists.

The SEL is equipped with a function for the update of time and a function for the update of the state of the CTPN, namely the state equation law

$$x_{k+1} = f(x_k, u_k).$$

When an event is enabled and the firing delay time is expired, the event occurs and the corresponding transition fires. The events contained in the SEL correspond to transitions t_i, in the case of PNs, or to couples transition-color $t_i c_j$ for CTPNs. Speaking in the terms of the SEL structure, each new enabled transition of a PN corresponds to a new *feasible* event in the SEL. A conflict exists

when two or more enabled events cannot happen both, since the occurrence of one of them disables the other one. The resolution of such a conflict is necessary in order to let the evolution of the DES continue. When a transition of a PN wins a conflict, the events corresponding to the losing transitions are deleted from the SEL, and they become *infeasible* events.

Chapter 3

A Model for Materials Handling Systems

Typical modeling approaches for MHSs are layout-dependent. This means that physical components of a warehouse are modeled and that the model's structure strictly depends on a specific layout.

In this work a general and multi-level modeling approach for MHSs is proposed. The two main components of the modeling framework are a *formal model* and an *auxiliary process* as shown in Fig. 3.1.

Through a (formal) CTPN model, activities in a MHS are modeled, rather than the physical components of the system. In this way the model's structure doesn't vary depending on the system, and it is not required a great effort to obtain, even through automated procedures, the model of a specific MHS.

The real customization of the model is devolved to the high level part of the modeling framework, i.e. the auxiliary process, which holds the high level information and the physical constraints proper of each specific plant.

In this chapter the CTPN model for MHSs and the auxiliary process are described more in depth.

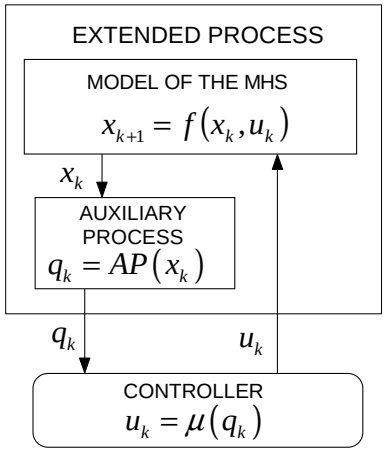

Figure 3.1 Extended Process: Formal model of the MHS and Auxiliary Process.

3.1 Formal model of the Process

3.1.1 CTPN Model of the Process

The basic element of the CTPN model is the mission. In MHSs, a mission consists in the transfer of a load from a location of a warehouse to another (an example of typical warehouse layout is shown in Fig. 3.2). A mission can be synthetically described in the following way:

$$move[ORIGIN, DESTINATION].$$

Each mission is here treated as a couple of *Elementary Missions*, i.e. a *Pickup* from an $ORIGIN$ and a *Drop-off* into a $DESTINATION$. Elementary missions are further fragmented in two parts: a *Goto Action* and a *Pickup/Drop-off Action*. Thus, to define each mission it is required to compute four time durations that strictly depend also on which resource (and its status) executes the mission:

1. the time required by the resource to reach the ORIGIN starting from the actual position;

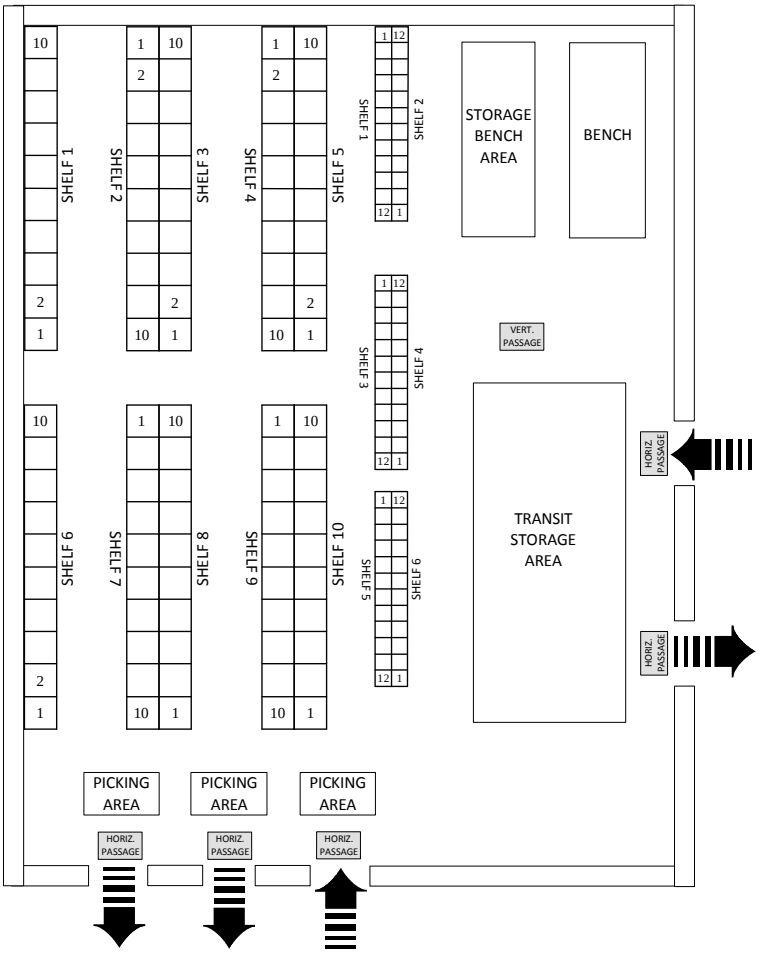

Figure 3.2 Example of physical layout of a MHS.

2. the time required by the resource to pick the load;

3. the time required by the resource to reach the DESTINATION;

4. the time required by the resource to drop-off the load.

In the CTPN framework each elementary pick mission has been modeled as a CTPN module made of a chain of places and (timed) transitions. In Fig. 3.3 a model of MHS with four missions and two resources is shown.

In Fig. 3.3 the transitions t_1, t_2 and t_3 respectively model:

1. the fact that the mission M_1 is begun;

2. the fact that the goto action has been completed;

3. the fact that the pickup action has been completed.

Elementary drop-off missions are instead modeled as place-transition chains connected to the resource places. This is due to the fact that drop-offs are defined at runtime. The presence of an arc which connects a chain (of places and transitions) representing the resource R_j and another one representing the mission M_i means that R_j and M_i are compatible.

The CTPN model of the process is obtained through an automated procedure which, starting from a buffer of missions and a list of active resources, builds the chains of places and transitions. The CTPN parameters, i.e. the time durations of transitions, depend on the choices made by the controller and thus, they are computed in run time. For instance:

- the time required by a resource to reach a location of the warehouse starting from the actual position is computed by dividing the distance to be traveled by the speed of the resource (see the auxiliary process). The travel distance is computed from an algorithm which finds, through and heuristic approach, the best (minimum) path among two locations of the warehouse;

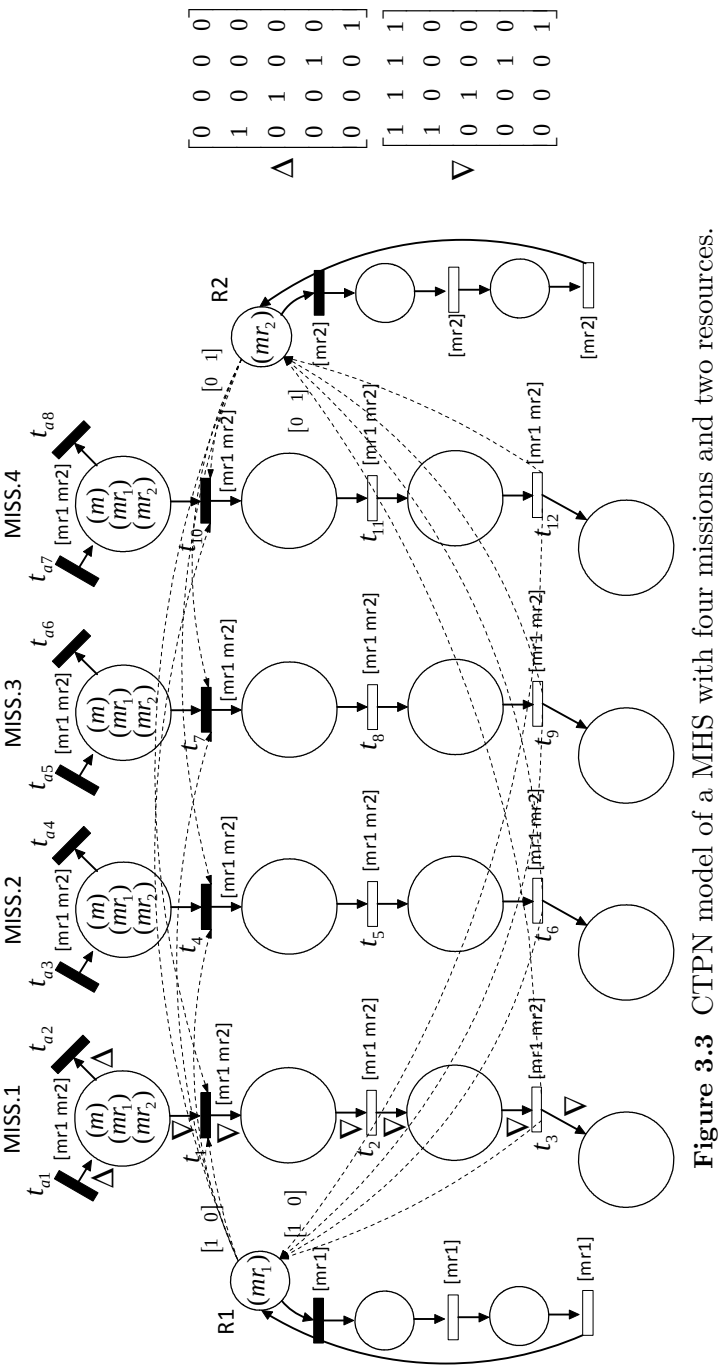

Figure 3.3 CTPN model of a MHS with four missions and two resources.

- the time required by a resource to pick or drop-off a load is computed by exploiting the high level information made available by the auxiliary process.

By selecting the firing color of a mission's transitions, the controller carries out its control action. After that all the controller's decisions have been taken, the CTPN autonomously evolves and, according to the CTPNs state equation, a new state is reached. The set of active missions has a biunique correspondence with the set of enable transitions of the CTPN.

The interface between the CTPN model and what lies above it containing the high level information (the auxiliary process which will be more deeply described in the following paragraph), is realized through some special transitions, also called "Auxiliary transitions" (labeled t_{a1}, t_{a2}, etc... in Fig. 3.3). From the point of view of the PNs semantics, the auxiliary transitions are untimed, and thus immediate, transitions that correspond to some auxiliary events. Moreover, when they are involved in conflicts, they have priority over the other transitions and they win the conflict. The auxiliary transitions regulate the presence of tokens into the first place of each mission in the following way:

- If M_1 has to be inhibited by the AP under a certain resource, the corresponding token is removed from its first place, through the firing of t_{a2};
- If M_1 is inhibited but it can be executed by a certain resource again, the corresponding token is added into its first place, through the firing of t_{a1}.

Remark: In this work mono server semantic (each transition can fire once at a time) is assumed since each resource can perform just one activity at a time.

3.1.2 Why CTPNs instead of PNs?

The use of CTPNs is due to their capacity to compress modular and similar components of the net. In CTPNs, indeed, each token

is a string whose symbols belong to a set of predefined symbols and which has got a coded meaning. In this context these symbols are integer numbers and represent the presence of a specific resource into a place of the CTPN. Through CTPNs it is possible to represent in a more compact manner the "OR" construct, as explained in the following. In Fig. 3.4 it is shown a (non-colored) PN where the first transition of a mission is connected to three resource places. According to the basic firing mechanism shown in Fig. 3.4, the tokens contained in all the places before the transition are taken away, when the transition t_1 fires. So it is not possible to choose only a subset of the places from which to take away the token.

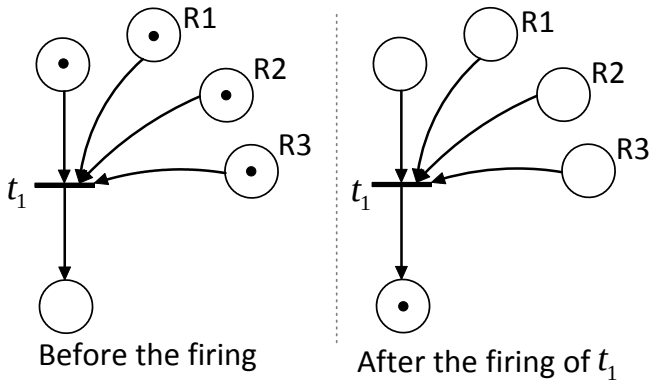

Figure 3.4 Petri nets firing mechanism.

This could be obtained by splitting the mission, as in Fig. 3.5. Then the controller could let one of the three clone transitions fire, depending on which resource has been chosen for the mission.

With CTPNs it is possible to compress this structure into a unique mission whose transitions can fire under a number of colors ($mr1$, $mr2$, $mr3$), equal to the number of connected (resource) places, like shown in Fig. 3.6.

As already highlighted, in the CTPN representation used in Fig. 3.6 and in the rest of this work the incidence matrix's entries are matrices. For the sake of brevity, this notation is not recalled here. For further details, the reader can refer to [BCC07].

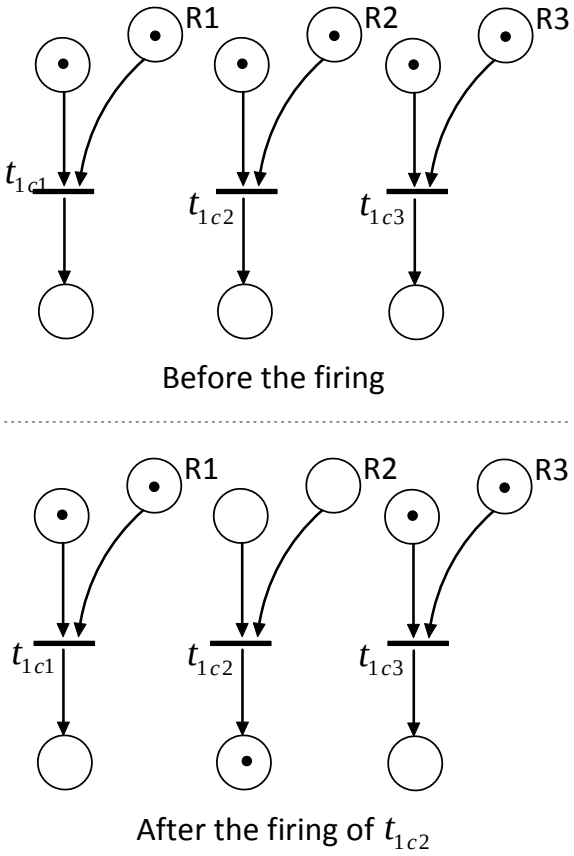

Figure 3.5 Splitting of a module in three clones.

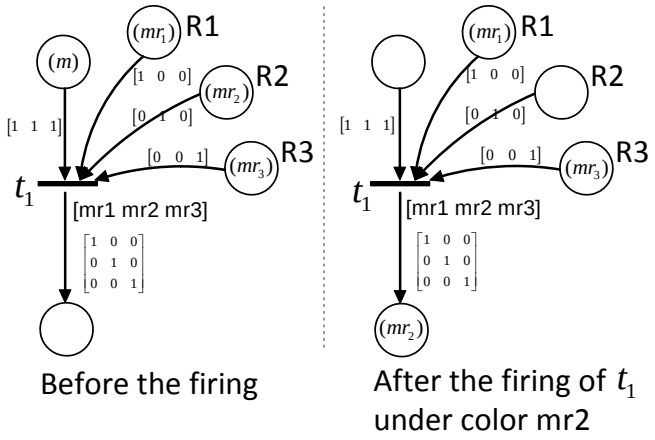

Figure 3.6 Colored Petri nets firing mechanism.

3.2 Auxiliary Process

What differentiates two MHSs is not the structure of the missions which always consist in a pickup and a drop-off, but the high level information which is specialized for each MHS. It has been decided to separate this platform-dependent and variable information from the formal model of the process to obtain a model structure as general as possible. Thus, the AP is necessary:

- To extend the CTPN, which models the process, with the high level and plant-dependent information that is not convenient to include in the CTPN.
- To express the high level constraints that cannot be defined through the CTPN constructs. This implicitly yields a minimization of the number of the choices to be made by the controller. For example the fact that a resource is full, makes the resource unable to accept other missions until it is emptied. When this condition is true, it does not make sense to take into account in the control decisions, the full resource.

The AP acts in two phases. In the "Update" phase it updates the high level information and the additional data structures, after

a firing. In the "Evaluate" phase it evaluates the logic conditions, based on the actual state of the system.

If necessary, namely when a logic condition is verified, the AP generates a *Logic Event* (LE). LEs are associated to the auxiliary transitions t_{ai} of Fig. 3.3.

The process evolves and generates *Process Events* (PEs) which correspond to the physical events like the completion of a task for a resource. PEs yield the firing of the standard transitions and the modification of the state of the CTPN model of the process. PEs allow the CTPN to interact with the AP which can read the updated marking of the CTPN useful for the update and the evaluate phases;

LEs model the occurrence of logic conditions, as the violation of physical constraints (the saturation of a resource for example) yielding the firing of the auxiliary transitions. Notice that the verification of a LE, i.e. the firing of an auxiliary transition reduces the set of missions to be considered when a control decision has to be made.

With respect to the modeling architecture shown in Fig. 3.1, the AP evaluates the actual state x_k of the MHS and logic conditions, and computes the extended process state

$$q_k = AP(x_k)$$

which can have got a smaller number of conflicts to be solved. Indeed, at each simulation step, the actual conflicts set is obtained after the action of the AP which modifies the net marking generating LEs. For a mission that has to be inhibited by the AP under a certain resource, the corresponding token is removed from the first place of the mission. The remaining enabled firing colors of a transition correspond to the resources that are physically able to execute a mission.

In order to demonstrate how constraints can be easily expressed in a formal way, an example is now proposed.

3.2.1 Example of high level information

Typical information of a MHS related to missions and resources, which can be used for both the definition of constraints and of dispatching rules, is here exemplified.

At first, consider the following mission-related high level information:

- M_h: h-th mission;
- $M_h.Identifier$: Identifier of the mission;
- $M_h.Index$: Index of the mission into the buffer;
- $M_h.Priority$: Priority of the mission;
- $M_h.Origin$: Origin location for the mission M_h. It is the first position to be reached by a resource when it executes the mission M_h;
- $M_h.Origin.Level$: Height of the ORIGIN for the mission M_h;
- $M_h.Destination$: Destination location for the mission M_h. It is the last position to be reached by a resource when it executes the mission M_h;
- $M_h.ItemWeight$: Weight of the part;
- $M_h.ItemVolume$: Volume of the part;
- $M_h.ItemFragility$: Fragility index of the part;
- $M_h.DueDate$: Due date of the mission;
- $M_h.EndTime$: Ending time of the mission;

Then, consider the following resource-related high level information:

- R_k: k-th resource;

- $R_k.Identifier$: Identifier of the resource;
- $R_k.ActualPos$: Actual position of the resource R_k;
- $R_k.Speed$: Cruise speed of the resource R_k;
- $R_k.Forbidden$: Forbidden locations for the resource.
- $R_k.LevelChangeTime$: Time required by the resource R_k to change level;
- $R_k.PickTime$: Time required by the resource R_k to pick the part from its location.
- $R_k.Height$: Reachable height of the resource;
- $R_k.MaxWeight$: Maximum weight transportable by the resource R_k;
- $R_k.MaxVolume$: Maximum volume transportable by the resource R_k;
- $R_k.FSV.Weight$: Actual total weight of parts onboard the resource R_k, from a filling status vector of the resource;
- $R_k.FSV.Volume$: Actual total volume of parts onboard the resource R_k.
- $R_k.EndTime$: Ending time of the last activity of the resource R_k.
- $R_k.ResWeight$: Residual transportable weight of the resource;
- $R_k.ResVolume$: Residual transportable volume of the resource;

Consider also some further notations:

- $NMiss$: Number of missions to be executed;

- $NMissR_k$: Number of missions that are linked to the resource R_k;
- $NFreeRes$: Number of available resources (in a certain time instant);
- $NFreeResM_h$: Number of available resources (in a certain time instant), for a specific mission M_h;
- $P(R_k)$: Place of the CTPN corresponding to the k-th resource;
- $m\,(P(R_k), t)$: Marking of the CTPNs place corresponding to the resource R_k at time t.

3.2.2 Examples of constraint

The information presented in the previous paragraph consists in scalar values or sets of scalar values. Thus, the expression of constraints is done through the relational operators:

$$=, +, -, \times, \div, <, \leq, >, \geq$$

and

$$\subset, \subseteq, \supset, \supseteq, \in, \cap, \cup$$

which are composed through the boolean operators AND, OR, NOT.

Let us suppose that the mission M_1 must not be executed by the resource R_1 if the available weight or volume of R_1 are not sufficient for the item of mission M_1. This constraint can be formalized by using the high level information of Paragraph 3.2.1:

$$\begin{aligned}[(M_1.ItemWeight \geq R_1.ResWeight)\ \text{OR}\\ (M_1.ItemVolume \geq R_1.ResVolume)]\end{aligned}$$

If this condition is true, the auxiliary transition that disables M_1 under R_1 fires.

In addition it could be imposed the disabling of M_1 under R_1 if the ORIGIN or the DESTINATION of the mission M_1 are forbidden for the resource R_1. In this case the constraint is the following:

$$[(M_1.Origin \subseteq R_1.Forbidden) \text{ OR } (M_1.Destination \subseteq R_1.Forbidden)]$$

Chapter 4

Simulation-Based Control

In a running MHS, the resolution of conflicts is necessary in order to let the evolution of the system advance. In the proposed modeling and control framework, the simulation is used, by the control environment, to determine when decisions have to be made, namely when conflicts occur. A conflict, for example, occurs when a resource is free and it has to be decided if the resource must execute a mission (and which one), proceed with an emptying action or do nothing. In this work it has been chosen to solve conflicts through the use of *Dispatching rules*.

In this chapter the devised *model-based control environment*, including the model simulation and the controller, is described and a survey of the most usual dispatching rules for MHSs is provided. Then, an *operational control scheme*, based on the model-based control environment, will be presented. Finally, on-line MHSs control issues will be discussed. In particular the possibility to have reschedules of the activities, based on a reconfigured model when some perturbations occur, will be considered.

4.1 Model-Based Control Environment

The model-based control environment designed for the detection and the solution of conflicts and confusions is shown in Fig. 4.1. It includes the CTPN model of the process, the auxiliary process

and the controller.

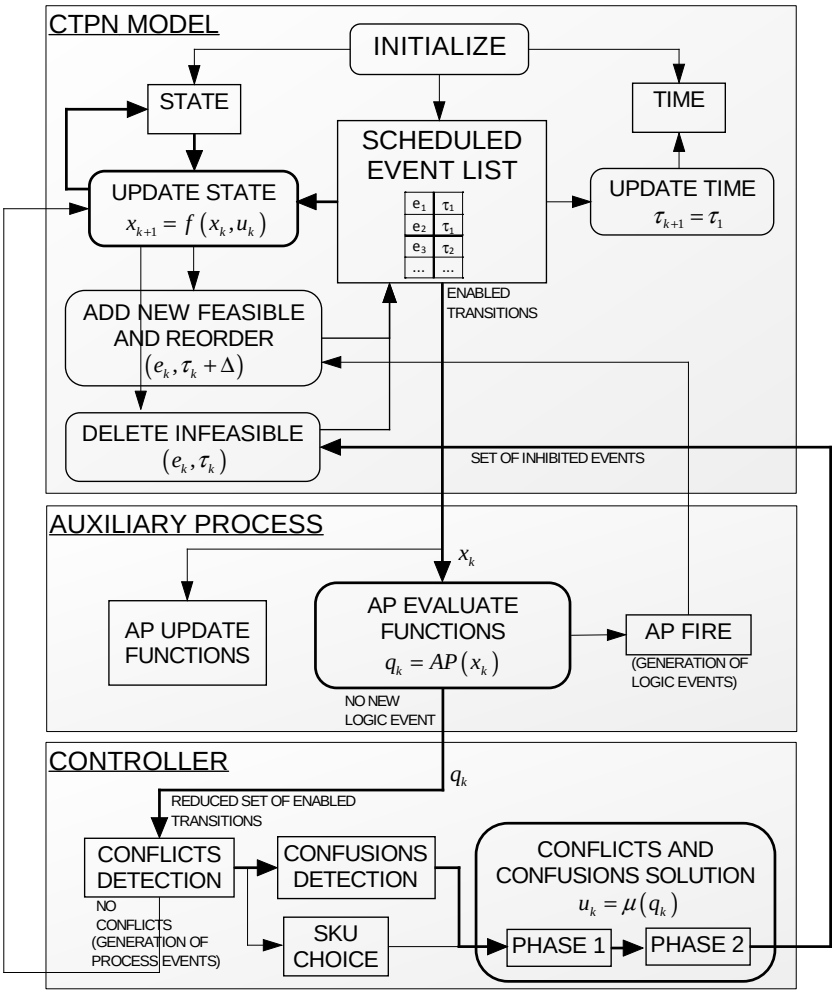

Figure 4.1 Model-Based Control Environment.

Given a buffer of missions to be scheduled and a set of available resources the main control problem of MHSs consists in determining which resource has to execute which mission. At each decision moment there can be a set of resources which is able to execute one or more missions. This reflects in the presence of several enabled transitions in the SEL (Paragraph 2.3). Through the simulation of

the CTPN model it is completely known when a control decision has to be made and who is involved in it.

In the model-based control environment of Fig. 4.1 the main role of the control module is to make decisions by solving conflicts according to a certain criterion, in a manner such that an objective function is optimized. The controller's action

$$u_k = \mu(q_k)$$

always follows the actual state q_k evaluation (see Paragraph 3.2).

Once conflicts have been solved and a mission is assigned to each resource, time durations of each mission's transitions can be computed and assigned. At this point the model evolves autonomously according to the state equation law, yielding a new state.

The inputs of the control problem are the following:

- Information on the physical system;
- Information on the resources;
- Buffer of missions to be executed.

When all the conflicts and the entire control problem have been solved, as an output a *Joblist* is generated for each resource. Thus, the outputs of the control system are the following:

- Execution time of the buffer of missions;
- Joblists for all the resources.

4.1.1 Algorithm for the Detection of Conflicts and Confusions

The algorithm designed for the detection of conflicts and confusion in a CTPN is now presented.

Algorithm
Require: CTPN model, SEL

1 *Build the input places sets for each transition (*$^{\cdot}t_ic_j$*)*:

$\forall i, j$ find all the places P_hC_k which have an outgoing arc towards t_ic_j with a weights matrix different from zero;

2 *Build the groups of conflicts*:

2.1 Compute the intersections $^{\cdot}t_ic_j \cap ^{\cdot} t_mc_n$ with $j \neq n$ if $i = m$;

2.2 **If** a new element into an intersection is found

Then a structural conflict has been detected: create a new group of conflicts;

3 *Detect structural confusions*:

If for a transition t_ic_j the set of input places $^{\cdot}t_ic_j$ contains elements that are part of at least two distinct groups

Then t_ic_j is in structural confusion;

4 *Detect behavioral confusions*:

If, at runtime, $\forall i, j$:

4.1 t_ic_j is in structural confusion **AND**

4.2 t_ic_j is enabled **AND**

4.3 for at least two groups of conflicts which t_ic_j is part of, already one transition $t_mc_n \neq t_ic_j$ is enabled

Then t_ic_j is in behavioral confusion.

Given a CTPN made of m places (with at most h colors) and n transitions (with at most k colors), the computational complexity of the algorithm for the detection of conflicts and confusions mainly depends on:

1. the building of at most $n \times k$ sets of places which have an outgoing arc towards each transition ($^{\cdot}t_ic_j$, $\forall i = 1, ..., N$ and $\forall j = 1, ..., k$);

2. the computational complexity of finding the

$$\sum_{i=1}^{n\times k-1} (n \times k - i)$$

intersections two by two among the $n \times k$ sets. Notice that the computational complexity of computing the intersection among two sets of p and q elements respectively is $O(pq)$ in the worst case, i.e. when evaluating element by element.

In order to show how the algorithm works, the CTPN of Fig. 2.4 is taken into account as an example. As the reader can see, in the CTPN of Fig. 2.4 the transitions t_2mr1 and t_2mr2 are in structural confusion. In the first step of the algorithm, the input places sets for all the transitions and all the colors are obtained like in (4.1):

$$\begin{array}{l} {}^{\cdot}t_1mr1 = \{M_1m, M_1mr1, R_1mr1\} \\ {}^{\cdot}t_2mr1 = \{R_1mr1, M_2m, M_2mr1\} \\ {}^{\cdot}t_2mr2 = \{M_2m, M_2mr2, R_2mr2\} \\ {}^{\cdot}t_3mr2 = \{R_2mr2, M_3m, M_3mr2\} \\ {}^{\cdot}t_{G1}mr1 = \{R_1mr1\} \\ {}^{\cdot}t_{G2}mr2 = \{R_2mr2\} \end{array} \tag{4.1}$$

Applying the second step of the algorithm, all the intersections ${}^{\cdot}t_ic_j \cap {}^{\cdot} t_mc_n$ computed yield the building of the three groups of conflicts $GR_1 : R_1mr1$, $GR_2 : M_2m$ and $GR_3 : R_2mr2$ like shown in the equations from (4.2) to (4.6):

$$\begin{array}{l} {}^{\cdot}t_1mr1 \cap {}^{\cdot} t_2mr1 = R_1mr1 \\ {}^{\cdot}t_1mr1 \cap {}^{\cdot} t_2mr2 = \emptyset \\ {}^{\cdot}t_1mr1 \cap {}^{\cdot} t_3mr2 = \emptyset \\ {}^{\cdot}t_1mr1 \cap {}^{\cdot} t_{G1}mr1 = \{R_1mr1\} \\ {}^{\cdot}t_1mr1 \cap {}^{\cdot} t_{G2}mr2 = \emptyset \end{array} \quad \Rightarrow \quad GR_1 : R_1mr1 \tag{4.2}$$

$$\begin{array}{l} {}^{\cdot}t_2mr1 \cap {}^{\cdot} t_2mr2 = \{M_2m\} \\ {}^{\cdot}t_2mr1 \cap {}^{\cdot} t_3mr2 = \emptyset \\ {}^{\cdot}t_2mr1 \cap {}^{\cdot} t_{G1}mr1 = \emptyset \\ {}^{\cdot}t_2mr1 \cap {}^{\cdot} t_{G2}mr2 = \emptyset \end{array} \quad \Rightarrow \quad GR_2 : M_2m \tag{4.3}$$

$$\begin{array}{l} {}^{\cdot}t_2mr2 \cap^{\cdot} t_3mr2 = \{R_2mr2\} \\ {}^{\cdot}t_2mr2 \cap^{\cdot} t_{G1}mr1 = \emptyset \\ {}^{\cdot}t_2mr2 \cap^{\cdot} t_{G2}mr2 = \{R_2mr2\} \end{array} \quad \Rightarrow \quad GR_3 : R_2mr2 \tag{4.4}$$

$$\begin{array}{l} {}^{\cdot}t_3mr2 \cap^{\cdot} t_{G1}mr1 = \emptyset \\ {}^{\cdot}t_3mr2 \cap^{\cdot} t_{G2}mr2 = \emptyset \end{array} \tag{4.5}$$

$${}^{\cdot}t_{G1}mr1 \cap^{\cdot} t_{G2}mr2 = \emptyset \tag{4.6}$$

In the third step the structural confusions are finally detected. The transitions t_2mr1 and t_2mr2 are found to have input places that belong to at least two distinct groups of conflicts, like shown in (4.7) and (4.8).

$$\begin{array}{l} {}^{\cdot}t_1mr1 = \{M_1m, M_1mr1, R_1mr1\} \\ {}^{\cdot}t_2mr1 = \{\mathbf{R_1mr1}, \mathbf{M_2m}, M_2mr1\} \\ {}^{\cdot}t_2mr2 = \{\mathbf{M_2m}, M_2mr2, \mathbf{R_2mr2}\} \\ {}^{\cdot}t_3mr2 = \{R_2mr2, M_3m, M_3mr2\} \\ {}^{\cdot}t_{G1}mr1 = \{R_1mr1\} \\ {}^{\cdot}t_{G2}mr2 = \{R_2mr2\} \end{array} \tag{4.7}$$

$$\begin{array}{l} {}^{\cdot}t_1mr1 \subseteq GR_1 \\ {}^{\cdot}\mathbf{t_2mr1} \subseteq \mathbf{GR_1}, \mathbf{GR_2} \\ {}^{\cdot}\mathbf{t_2mr2} \subseteq \mathbf{GR_2}, \mathbf{GR_3} \\ {}^{\cdot}t_3mr2 \subseteq GR_3 \\ {}^{\cdot}t_{G1}mr1 \subseteq GR_1 \\ {}^{\cdot}t_{G2}mr2 \subseteq GR_3 \end{array} \tag{4.8}$$

Thus t_2mr1 and t_2mr2 are in structural confusion like supposed. For the sake of brevity, the runtime simulation of the example, required to detect also behavioral confusions, is not reported in this work. However, it can be said that t_2mr1 and t_2mr2 are in behavioral confusion if, for example, $t_{G1}mr1$, t_2mr1, t_2mr2 and $t_{G2}mr2$ are all enabled.

4.1.2 Two Phase Control Action

In order to solve the confusions, it is here proposed the two phases control action shown in Fig. 4.2. At first (phase 1) the best re-

source R_i for each mission M_j is selected, according to a dispatching rule. Confusions are so avoided. Then, conflicts are solved (phase 2) by assigning the best mission to each resource, according to another dispatching rule.

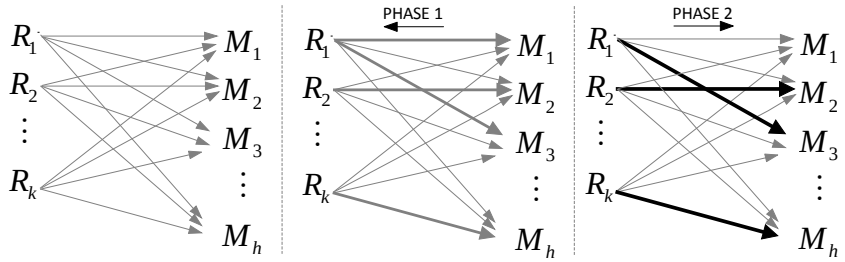

Figure 4.2 Two phases action of the controller.

The use of dispatching rules, rather than the design of complex algorithms, could result, first of all, in a more reasonable way to face the possible combinatorial explosion of this mission-to-resource assignment problem. Moreover, it is important to notice that the solution of a dispatching problem also depends on the specific input buffer of missions and that the mission arrival process is often very turbulent. Thus, although the solution of the control problem via dispatching rules produces only locally optimal solutions, to find a closed analytical solution for global planning does not make sense [GZNL08] if the sudden arrival of a new mission could require a complete re-schedule of the work.

In this paragraph the control policy μ is assumed constant and statically chosen among a set of dispatching rules. In Paragraph 4.4 the control scheme of Fig. 4.1 will be englobed into a higher level architecture where μ will be let vary for each batch of missions. The control architecture of Fig. 4.1 will be used to simulate the system and to evaluate several dispatching rules on each batch of missions, in order to always choose the one that fits better with the considered batch.

4.2 Objective Functions (OFs)

One of the most important objectives to be reached in real MHSs is the minimization of the execution time of the missions. A typical time-performance measure, for the activities of a system, is the so-called "Makespan" (MS). Given a set of resources and assumed that they have completed their joblists, the MS is the last ending time among all the resources:

$$MS = \max_{k=1}^{NumRes} (R_k.EndTime)$$

In contexts where missions have very different time durations it can happen that there is one last resource involved in a long mission, while quite all the resources have completed their jobs, thus being available for new jobs. In this case the MS behaves like non-balanced measure thus not capturing the goodness of a control action. For this reason the Average of the Ending Times (AET) of resources, has also been introduced which captures the quality of a control action more effectively than the MS. The AET is defined as follows:

$$AET = \frac{1}{NumRes} \cdot \sum_{k=1}^{NumRes} (R_k.EndTime)$$

As highlighted in [WW89], the use of fixed control actions, which minimize the execution time of activities, can produce late missions. When using time-minimizing dispatching rules, in order to avoid the so-called *Starvation* phenomenon, the Number of Late Missions (NLM) can be also taken into account. NLM consists in the number of missions that ended after their due date, i.e. the missions for which $M_h.EndTime > M_h.DueDate$ and it has been considered to characterize the lateness in a MHS.

When several objectives have to be reached at the same time, a good compromise could be to use a multi-objective Objective Function (OF). For example, calling

$$x_1 = MS,$$

$$x_2 = AET$$

and

$$x_3 = NLM$$

the objective could be to minimize

$$z = c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3.$$

The main drawback of using a multi-objective OF is the difficulty to tune the weights c_i of the single components. In literature several techniques, like evolutionary and genetic algorithms, have been proposed which, however, can be applied to systems not as time variant as the ones here considered. Indeed, when the system's characteristics are too time-dependent, sophisticated tuning sets can be rapidly made vain. When simulation is available, it can be used for a trial-and-error tuning technique based on the observation of simulation results.

4.3 Dispatching Rules

In this Paragraph the most typical dispatching rules for MHSs are presented and formally described exploiting the CTPN constructs and the high level information made available by the auxiliary process.

In general dispatching rules can be referred to resources, to missions or to both of them. Thus, they can be applied in phase 1, phase 2 or both phases of the control action. In Fig.4.3, nearby the acronyms used in the rest of the work to denote the rules, it is specified in which phase each rule is applicable.

In Paragraph 4.3.17 a simulation comparison among dispatching rules is presented. The rules are compared from the point of view of the makespan and of the average execution time and they are tested on the same example made by 6 resources and 150 missions.

Description	Acronym	Phase1	Phase2
RND	Random	✓	✓
MD	Minimum Distance	✓	✓
MMT	Minimum Moving Time	✓	✓
MPT	Minimum Pick Time	✓	✓
MTPT	Minimum Total Processing Time	✓	✓
ER	Emptiest Resource	✓	
FR	Fullest Resource	✓	
LUR	Least Utilized Resource	✓	
MUR	Most Utilized Resource	✓	
LIR	Longest Idle Resource	✓	
MRAR	Most Recently Active Resource	✓	
MS	Max Speed	✓	
FCFS	First Come First Served Mission		✓
PRI	Highest Priority Mission		✓
SIF	Smallest Item Mission First		✓
BIF	Biggest Item Mission First		✓

Figure 4.3 Typical dispatching rules in MHSs.

4.3.1 Random Rule

The "RND" rule chooses a resource and a mission at random. It consists in assigning missions to resources with no optimization criterion and it also corresponds to what it is actually done in non-controlled real MHSs. The RND rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = Random\,(1, ..., NFreeResM_h)$$

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = Random\,(1, ..., NMissR_k)$$

4.3.2 Minimum Distance Rule

The "MD" rule chooses the resource and the mission which are closest. It can be used to reduce the total traveled distance of

resources thus reducing the need of maintenance. The MD rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} (distance\,(R_i.ActualPos, M_h.OriginSKU))$$

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \min_{j=1,...,NMissR_k} (distance\,(R_k.ActualPos, M_j.OriginSKU))$$

4.3.3 Minimum Moving Time Rule

The "MMT" rule chooses the resource and the mission for which the travel time of the resource is minimum. In addition to the rule MD, it considers also resources speeds. The MMT rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} \left(\frac{distance\,(R_i.ActualPos, M_h.OriginSKU)}{R_i.Speed} \right)$$

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \min_{j=1,...,NMissR_k} \left(\frac{distance\,(R_k.ActualPos, M_j.OriginSKU)}{R_k.Speed} \right)$$

4.3.4 Minimum Pick Time Rule

The "MPT" rule chooses the resource and the mission for which the pick time of the part is minimum. The MPT rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} (2 \times M_h.OriginSKU.Level \times R_i.LevelChangeTime + R_i.PickTime)$$

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \min_{j=1,...,NMissR_k} (2 \times M_j.OriginSKU.Level$$

$$\times R_k.LevelChangeTime + R_k.PickTime)$$

4.3.5 Minimum Total Processing Time Rule

The "MTPT" rule chooses the resource and the mission for which the sum of the travel time and of the pick time is minimum. This rule should generally provide the best results in terms of average ending time. The MTPT rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} (\frac{distance\,(R_i.ActualPos, M_h.Origin)}{R_i.Speed} +$$

$$+2 \times M_h.Origin.Level \times R_i.LevelChangeTime + R_i.PickTime)$$

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \min_{j=1,...,NMissR_k} (\frac{distance\,(R_k.ActualPos, M_j.Origin)}{R_k.Speed} +$$

$$+2 \times M_j.Origin.Level \times R_k.LevelChangeTime + R_k.PickTime)$$

4.3.6 Emptiest Resource Rule

The "ER" rule chooses in phase 1 the emptiest resource. This yields a uniform filling and a uniform use of resources. The ER rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} \left(\max \left\{ \frac{R_i.FSV.Weight}{R_i.MaxWeight}, \frac{R_i.FSV.Volume}{R_i.MaxVolume} \right\} \right)$$

4.3.7 Fullest Resource Rule

The "FR" rule chooses in phase 1 the fullest resource. This yields a heterogeneous filling and use of resources. The FR rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \max_{i=1,...,NFreeResM_h} \left(\max \left\{ \frac{R_i.FSV.Weight}{R_i.MaxWeight}, \frac{R_i.FSV.Volume}{R_i.MaxVolume} \right\} \right)$$

4.3.8 Least Utilized Resource Rule

The "LUR" rule chooses the resource that has been used for less time, yielding a uniform use of resources and a minimization of the number of idle resources. The LUR rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \max_{i=1,...,NFreeResM_h} \left(\int_0^t m\left(P(R_i), \tau\right) d\tau \right) =$$

$$= \max_{i=1,...,NFreeResM_h} \left(\sum_q \left(t_{qf} - t_{qi}\right) : \tau \in [t_{qi}, t_{qf}[\,, m\left(P(R_i), \tau\right) > 0 \right)$$

4.3.9 Most Utilized Resource Rule

The "MUR" rule chooses the resource that has been used for much time, yielding a non uniform use of resources and over-use of only some resources. The MUR rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} \left(\int_0^t m\left(P(R_i), \tau\right) d\tau \right) =$$

$$= \min_{i=1,...,NFreeResM_h} \left(\sum_q \left(t_{qf} - t_{qi}\right) : \tau \in [t_{qi}, t_{qf}[\,, m\left(P(R_i), \tau\right) > 0 \right)$$

4.3.10 Longest Idle Resource Rule

The "LIR" rule chooses the resource that has been used, the last time, longest ago. This yields a minimization of the number of idle resources. The LIR rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \min_{i=1,...,NFreeResM_h} \left(\max_t \left[m\left(P(R_i), t\right) > 0 \right] \right)$$

4.3.11 Most Recently Active Rule Rule

The "MRAR" rule chooses the resource that has been used, the last time, much recently. This yields a maximization of the number of idle resources. The MRAR rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \max_{i=1,...,NFreeResM_h} \left(\max_t \left[m\left(P(R_i), t\right) > 0 \right] \right)$$

4.3.12 Max Speed Rule

The "MS" rule chooses the fastest resource. This does not mean that the time required to reach the part is the shortest, since distance is not considered. The MS rule can be formally expressed as follows:

In phase 1 $\forall M_h : h = 1, ..., NMiss$ choose R_i:

$$i = \max_{i=1,...,NFreeResM_h} \left(R_i.Speed \right)$$

4.3.13 First Come First Served Mission Rule

The "FCFS" rule chooses the first available mission of the buffer. Missions are stored in the buffer in their arrival order, so that by choosing the first request arrived, the waiting time of missions will be minimized. The FCFS rule can be formally expressed as follows:

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \min_{j=1,...,NMissR_k} (M_j.Index)$$

4.3.14 Highest Priority Mission Rule

The "PRI" rule chooses the highest priority mission. This maximizes the respect of the due date of missions and limits the starvation phenomenon (it can be used in combination with the rule LUR for phase 1). The PRI rule can be formally expressed as follows:

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \max_{j=1,...,NMissR_k} (M_j.Priority)$$

4.3.15 Smallest Item Mission Rule

The "SIF" rule chooses in phase 2 the mission corresponding to the smallest part. This favors the complete filling of resources. The SIF rule can be formally expressed as follows:

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \min_{j=1,...,NMissR_k} \left(\max \left\{ \frac{M_j.Weight}{R_k.MaxWeight}, \frac{M_j.Volume}{R_k.MaxVolume} \right\} \right)$$

4.3.16 Biggest Item Mission Rule

The "BIF" rule chooses in phase 2 the mission corresponding to the biggest part. This yields an incomplete filling of resources, but let large parts be managed at first. The BIF rule can be formally expressed as follows:

In phase 2 $\forall R_k : k = 1, ..., NFreeRes$ choose M_j:

$$j = \max_{j=1,...,NMissR_k} \left(\max \left\{ \frac{M_j.Weight}{R_k.MaxWeight}, \frac{M_j.Volume}{R_k.MaxVolume} \right\} \right)$$

4.3.17 Dispatching Rules Comparison

A comparison among dispatching rules is here presented. The rules are compared from the point of view of the makespan and of the average execution time and they are tested on the same example of 6 resources and 150 missions.

Simulation results are shown in Fig. 4.4 where both the MS and the AET are depicted for each dispatching rule.

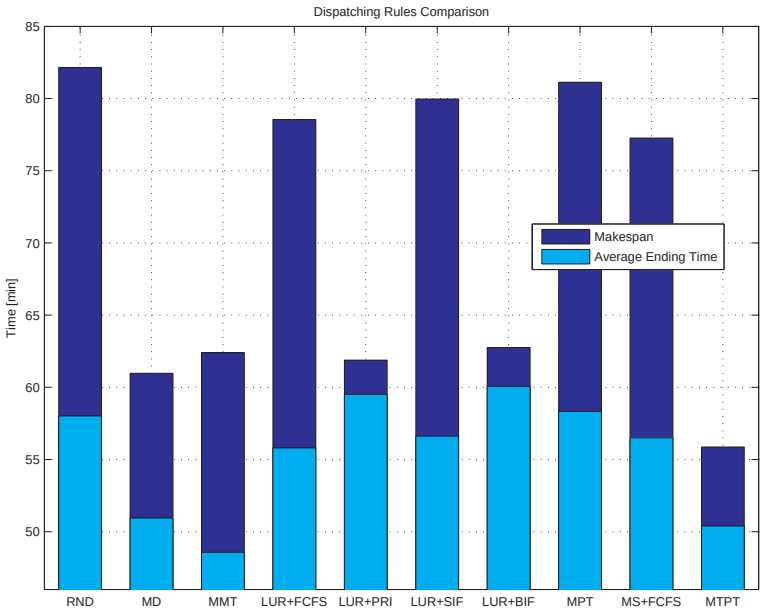

Figure 4.4 Comparison among the Dispatching Rules.

From Fig. 4.4 it can be observed that the best (minimum) AET is obtained with the rule MMT which allows, for example, to save about 10 minutes w.r.t. the rule PRI for the minimization of the number of late missions. The best (minimum) MS is obtained with the rule MTPT which allows to complete all the missions 26 minutes in advance w.r.t the RND rule.

The purpose of this test is, first of all, to show how certain rules directly influence, more than others, time performance measures of the system like MS and AET. Then, it is worth to observe how a

the rule which produces the best AET and the rule which produces the best MS do not necessarily have to coincide. Indeed, in this example they are respectively the MMT rule and the MTPT rule.

Moreover, depending on the input data, it can happen that a rule which is not expected to improve the time performances of the system, like the RND rule, produces better results than rules like MTPT which has got a direct influence on the MS. This happens just because the mechanism of the dispatching rules produces only local optimal solutions. The consequence is that which rule outperforms the others depends on the input data of the dispatching problem, i.e. on the specific buffer of missions.

Finally, it is important to underline that, from a computational point of view, the dispatching rules have proved to have almost the same requirements thus allowing to be used indifferently.

4.4 Operational Control Scheme

In the model-based control framework previously presented (Fig. 4.1) the control rule μ has been assumed fixed.

In this paragraph such control framework has been englobed into the higher level control architecture of Fig. 4.5 (the aforementioned *operational control scheme*), in which it has been assumed that the control rule could vary during the optimization process of several batches of missions. In the proposed operational control scheme the framework of Fig. 4.1 is used to simulate the system under one dispatching rule at time. It is so possible to test via simulation every dispatching rule in advance and choose the best one for each batch of missions.

The control architecture of Fig. 4.5 is also ready to go online with the system. Thanks to the formal model, system's simulation can be halted and recover when some disturbance occurs.

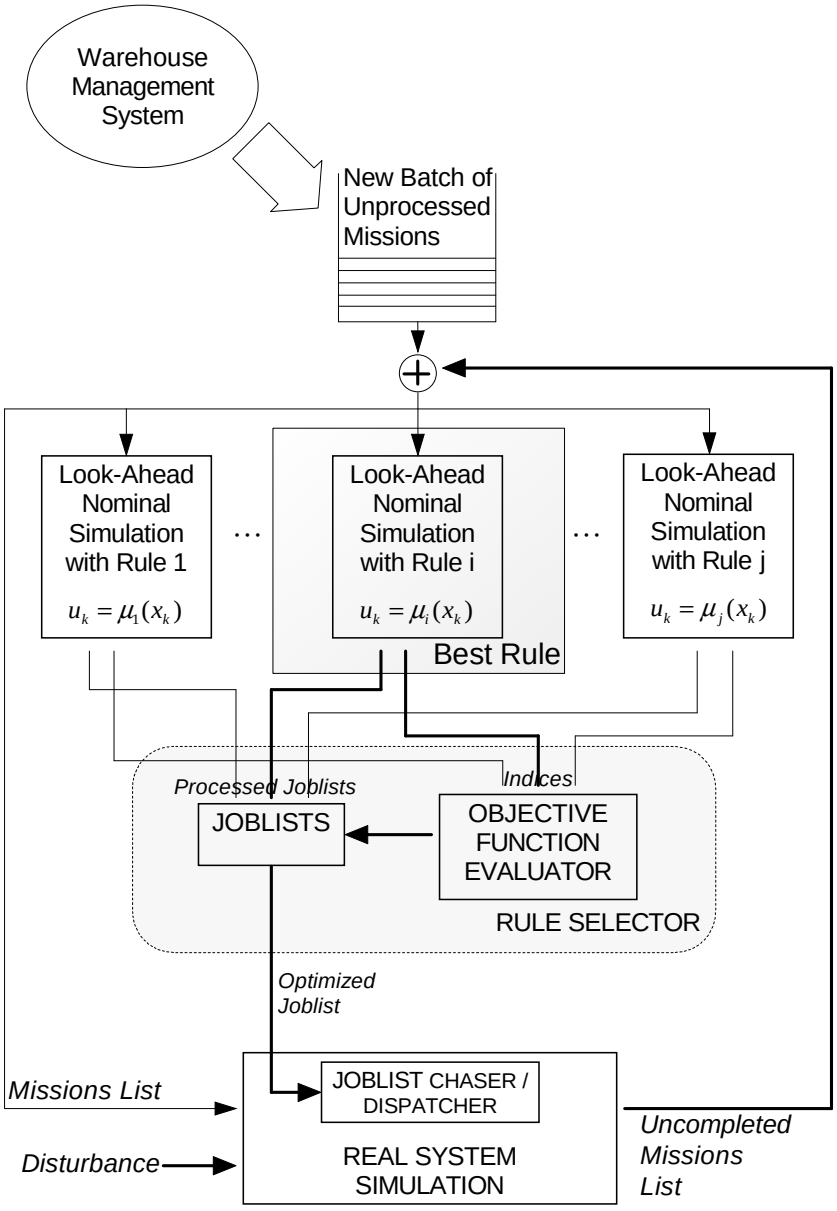

Figure 4.5 Operational control scheme.

4.4.1 Fixed Rule and Variable Rule Control Approaches

In the following chapter several simulation results will be presented both to deepen the comparison among the dispatching rules and to test the control architecture. Two simulation approaches will be used which have been called respectively "Fixed Rule Control Approach" and "Variable Rule Control Approach".

In the former approach, given a dispatching problem with more than one batch of missions to be optimized, the rule is supposed fixed for all the batches. This means that the operational control scheme of Fig. 4.5 is not fully exploited. Indeed rules performances are not measured and compared to choose the best rule, since rules are not varied.

In the latter approach the operational control scheme of Fig. 4.5 is fully exploited.

4.4.2 Online Control: Reaction to Disturbances

In real MHSs, many unforseen events, or disturbances, can occur which make a control problem different from the expected (nominal) one. Indeed, the arrival of missions is a strongly random and asynchronous process. Thus, for example, during the execution of a set of missions it can happen that one or more new urgent missions arrive which have to be included in the current schedule. Furthermore, there can be delays in the arrival of missions. As well as, from the point of view of the resources, it can happen that a resource breaks or that some new resources are added to the system. An off-line kind of control is usually realized in real MHSs. Indeed, whether the alteration is in the mission set or in the resource set, the schedules produced (off-line) before the occurrence of a disturbance are not altered and missions that cannot be executed are temporarily ignored. It is obvious that this is not a "reaction" to a disturbance since the activities are not rescheduled.

In this paragraph the use of the control architecture of Fig. 4.5

for the online control is discussed. The proposed architecture indeed allows the control system to react to disturbances by rescheduling the work in the respect to the new conditions.

In the scheme of Fig. 4.5 the unforseen events are represented by the input disturbance of the "real/simulated system" block. When a disturbance occurs the simulation of the real system is stopped, the system state is saved and the model is reconfigured. This is possible since the state of each mission is implicitly coded into the marking of the CTPN model which can be easily saved and restored. Then, on the basis of the reconfigured model, i.e. by taking into account the new system's conditions and inputs, the work can be rescheduled. As it will be shown from the simulation results of Chapter 5, the online control of the system is possible since this reaction process is fast enough to not stop the real system execution.

Kinds of Disturbances Considered

Two kinds of perturbations or disturbances, which can require a re-scheduling of the activities, are considered in this thesis:

1. Disturbances on the mission set:
   - *Missed Missions Arrival*;
   - *Delayed Missions Arrival*;
   - *Unexpected Urgent Missions Arrival*;
2. Disturbances on the resource set:
   - *Breakdown of a Resource*;
   - *Adding of a Resource*.

Reconfiguration of the Model

During the (online) simulation of a MHS, a mission's state can be *Waiting*, *In progress* or *Completed*. When the simulation is halted the *Waiting* missions are saved in an output buffer. For

all the *In progress* missions the current CTPN marking is saved into a proper data structure. On the new model building, the *In progress* and the *Waiting* missions are re-drawn and the state of the *In progress* ones is resumed by simply restoring their marking. If a new batch of missions is also available during this process, the new batch is appended to the output buffer as shown in Fig. 4.5. When re-building the model the adding of new available resources or the exclusion of broken ones, is also considered.

Reaction Strategies

In this thesis two reaction strategies to disturbances are proposed and compared (see Paragraph 4.4.2):

- Strategy 1 (*"Do what you can"*): this strategy corresponds to an off-line kind of control since the initial joblists are not altered. Unavailable missions are executed as soon as they become available again. The missions scheduled for a resource which is broken are executed as soon as possible, if possible, by the remaining resources.
- Strategy 2 (*Complete Rescheduling*): this strategy corresponds to the online control of a MHS. Indeed, joblists are nullified, the system's model is reconfigured and new joblists are produced in the respect of the new system's conditions.

Chapter 5

Simulation Results

In this chapter several simulations, inspired by a real case study, are presented to show the effectiveness of the proposed modeling and control approach. For the sake of brevity, too technical details on simulation data, like the properties of the resources, are omitted.

At first two simulation strategies (called batch simulation and single buffer simulation) are compared. Then a comparison among the dispatching rules is performed. Finally, the operational control scheme previously presented is fully exploited by fixing an objective function and letting the dispatching rule vary along a series of batches of missions. Also the online control in case of disturbances is tested and discussed.

All the simulations have been performed on an Intel Pentium IV 2.4 Ghz processor computer with 2 GB of RAM memory running Windows XP.

Remark: As it will be also clear from the simulations of this chapter, the outcome of the use of each dispatching rule depends on the input data of the control problem, i.e. on the set of missions considered. This means that two different batches having an equal number of missions can produce different simulation results (in terms, for example, of makespan) even if the dispatching rule is the same. Moreover, it is important to underline that the solution of each conflict with a dispatching rule is local. This implies that,

over a certain time horizon (or group of missions), the obtained control solution is not a global optimum. For instance, it can exist a combination of control decisions over a batch of missions which produces better results. Even this aspect will be clarified through simulations.

5.1 Fixed-Rule Simulations: Batch and Single Buffer Simulation

In this paragraph two different simulation strategies are discussed. The two strategies differ in the way missions are grouped to form the input buffer of the simulator. In real MHSs missions are typically grouped in small batches, which are requested during each day. This corresponds to what has been here called "batch simulation", in which the optimization of each batch of missions can be accomplished as soon as they become available. Theoretically, if all the missions of the day were available from the morning, the optimization of the whole set of missions could be accomplished. This would produce better optimization results. The simulation of a large set of missions at one go has been called "single buffer simulation".

Several simulations have been done to compare the two simulation strategies. Simulations have been accomplished considering a set of 6 resources and an increasing number of missions from 20 up to 200, given as single input buffer or as series of 20-missions batches. The dispatching rule has been fixed; which rule has been employed is not an aspect of interest in this paragraph.

From the point of view of the optimization results, as forecast, the results presented in Fig. 5.1 show that the single buffer simulation outperforms the batch simulation.

Indeed, for a problem of 100 missions, with the batch simulation, resources terminate their work 9 minutes later w.r.t. the 33 minutes of the single buffer simulation. These 9 minutes become 14 w.r.t the 69 minutes of the single buffer simulation in the case of 200 missions. The worsening of the batch simulation, grows

linearly with the number of missions.

From a computational point of view, as shown in Fig. 5.2, with the single buffer simulation the simulation time grows exponentially with the number of missions.

Number of Missions	SINGLE BUFFER SIMULATION		BATCH SIMULATION		Difference of the AET with Batch and Single Buffer Simulation (min)
	Simulation Time (sec)	AET (sec)	Simulation Time (sec)	AET (sec)	
20	9,781	522	9,797	522	0
40	18,094	880	19,612	1038	3
60	31,812	1309	30,532	1558	4
80	53,313	1967	39,234	2060	2
100	75,500	1974	48,938	2517	9
120	108,907	2362	59,231	2964	10
140	148,234	2816	68,828	3464	11
160	193,938	3105	77,766	4029	15
180	254,047	3492	86,360	4419	15
200	338,625	4160	95,985	4976	14

Figure 5.1 Simulation time with single buffer and batch simulation in presence of 6 resources and a variable number of missions.

About 6 minutes are necessary to simulate a dispatching problem with 200 missions (which correspond to a time horizon of about 75 minutes, in the given example), but it could become very hard to solve larger control problems in a reasonable time. With the batch simulation the simulation time grows linearly with the number of missions and less than 2 minutes are necessary to solve the same dispatching problem having 200 missions.

The simulation time considered in Fig. 5.1 and Fig. 5.2 is the sum of:

1. the time necessary, for the automated model building procedure, to compute the formal model;

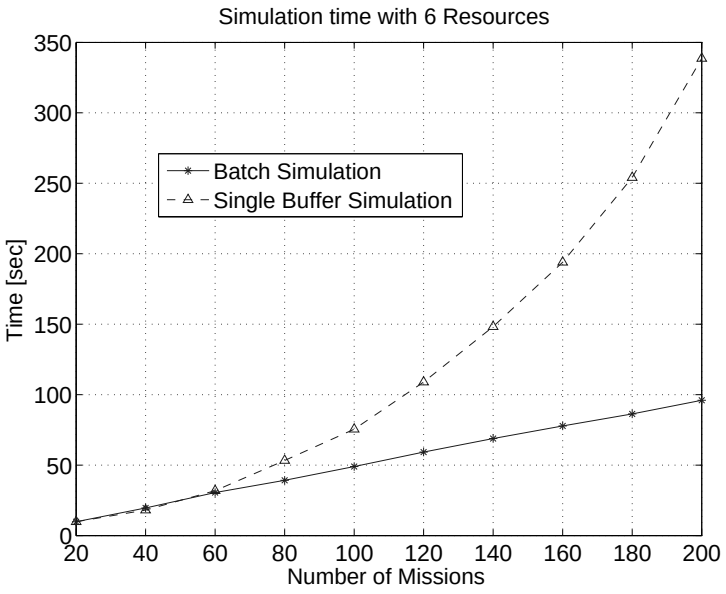

Figure 5.2 Simulation time with single buffer and batch simulation (graphical).

2. the time necessary to build the PN file in the standard Petri Net Markup Language (PNML) interchange format for PNs;

3. the time necessary to prepare the simulation engine (allocation of data structures and creation of input/output files);

4. the time necessary to complete the simulation.

It is important to notice that, however, the batch simulation has to be preferred to the single buffer simulation since it is more realistic and it has immediate applicability in real MHSs. Indeed, the only kind of MHSs in which the single buffer simulation is possible since the mission set of a day is known from the morning, are production warehouses where each day's plan is usually not subject to disturbances thus allowing to be scheduled in advance. For these reasons, from now on, only the batch simulation mode will be considered.

5.2 Fixed-Rule Simulations: Comparison Among Dispatching Rules

In this paragraph the simulation results of a case study consisting of 6 resources and 4 batches of 60 missions are presented. Only some of the dispatching rules presented in Chapter 4 are considered (in particular RND, MD, MMT, MPT, MTPT and LUR+PRI) which have been compared only from the points of view of the makespan and of the average ending time.

In all the simulations the rule is supposed fixed along the four batches. This means that objective functions are not used to detect the best rule to be chosen. Thus, the operational control scheme presented in Paragraph 4.4 is not yet fully exploited.

5.2.1 RND rule

Through the RND rule a resource and a mission at random are chosen in each conflict. This corresponds to what it is actually done in non-controlled real MHSs.

Rule Phase 1	Rule Phase 2	Batch	AET		MS	
			(sec)	(min)	(sec)	(min)
RND	RND	1	1525	25	1643	27
		2	1263	21	1326	22
		3	1438	24	2525	42
		4	1382	23	1540	26
		TOT	**5608**	**93**	**7034**	**117**

Figure 5.3 Fixed rule simulation results with RND rule.

As shown in Fig.5.3, with the random rule the total MS is 117 minutes and the total AET is 93 minutes.

These values will be used as a basis of comparison for the other rules: MS values close to 117 minutes or AET values close to 93 minutes will be considered “bad” behaviors. However, according to what has been remarked in the introduction of this chapter, even worse behaviors are possible.

Notice in the third batch of missions the difference between the ending time of the last resource (MS) and the average of the ending times (AET). This is an evident example of how the MS can result an unbalanced measure when mission durations differ much.

5.2.2 MD rule

Through the MD rule the resource and the mission which are closest are chosen. Only in case of resources having the same speed, this rule directly minimizes also the travel time of the resource.

Rule Phase 1	Rule Phase 2	Batch	AET		MS	
			(sec)	(min)	(sec)	(min)
MD	MD	1	1279	21	1467	24
		2	1173	20	1539	26
		3	1165	19	2165	36
		4	1149	19	1348	22
		TOT	**4766**	**79**	**6519**	**109**

Figure 5.4 Fixed rule simulation results with MD rule.

As shown in Fig.5.4, the total MS is 109 minutes and the total AET is 79 minutes. Thus, w.r.t. the RND rule, the MS is reduced (improved) by 7.3% and the AET is reduced (improved) by 15%.

With the MD rule the best AET is obtained.

5.2.3 MMT rule

Through the MMT rule the resource and the mission for which the travel time of the resource is minimum are chosen. The travel time is only part of the total time of a mission since also the pick/drop-off time has to be considered. However, in large warehouses, it represents the main contribution to the total processing time of a mission. In addition to the rule MD, with the MMT rule also resources speeds are considered.

As shown in Fig.5.5, the total MS is 108 minutes and the total AET is 80 minutes. Thus, w.r.t. the RND rule, the MS is reduced

Rule Phase 1	Rule Phase 2	Batch	AET		MS	
			(sec)	(min)	(sec)	(min)
MMT	MMT	1	1283	21	1442	24
		2	1204	20	1479	25
		3	1138	19	2212	37
		4	1158	19	1366	23
		TOT	**4783**	**80**	**6499**	**108**

Figure 5.5 Fixed rule simulation results with MMT rule.

(improved) by 7.7% and the AET is reduced (improved) by 14%.

With the MMT rule the best MS is obtained.

5.2.4 MPT rule

Through the MPT rule the resource and the mission, for which the pick/drop-off time of the part is minimum, is chosen. The pick/drop-off time is only part of the total time of a mission since also the travel time has to be considered. In large warehouses, it represents a minimum contribution to the total processing time of a mission.

Rule Phase 1	Rule Phase 2	Batch	AET		MS	
			(sec)	(min)	(sec)	(min)
MPT	MPT	1	1570	26	1691	28
		2	1347	22	1462	24
		3	1385	23	2537	42
		4	1252	21	1363	23
		TOT	**5554**	**93**	**7053**	**118**

Figure 5.6 Fixed rule simulation results with MPT rule.

As shown in Fig.5.6, the total MS is 118 minutes and the total AET is 93 minutes. Thus, w.r.t. the RND rule, the MS is not improved since it is augmented (worsened) by 0.9% and the AET is the same. This is an example of how a rule can have an unexpected behavior, on certain input data.

5.2.5 MTPT rule

Through the MTPT rule the resource and the mission, for which the sum of the travel time and of the pick time is minimum, are chosen. This rule should generally provide the best results in terms of average ending time.

Rule Phase 1	Rule Phase 2	Batch	AET		MS	
			(sec)	(min)	(sec)	(min)
MTPT	MTPT	1	1309	22	1615	27
		2	1152	19	1376	23
		3	1138	19	2212	37
		4	1134	19	1317	22
		TOT	**4733**	**79**	**6520**	**109**

Figure 5.7 Fixed rule simulation results with MTPT rule.

As shown in Fig.5.7, the total MS is 109 minutes and the total AET is 79 minutes. Thus, w.r.t. the RND rule, the MS is reduced (improved) by 6.8% and the AET is reduced (improved) by 15%.

With the MTPT rule the best AET is obtained.

5.2.6 PRI rule

Through the PRI rule the highest priority mission is chosen. This maximizes the respect of the due date of missions and limits the starvation phenomenon.

Rule Phase 1	Rule Phase 2	Batch	AET		MS	
			(sec)	(min)	(sec)	(min)
LUR	PRI	1	1516	25	1648	27
		2	1304	22	1391	23
		3	1443	24	2065	34
		4	1410	24	1527	25
		TOT	**5673**	**95**	**6631**	**111**

Figure 5.8 Fixed rule simulation results with PRI rule.

As shown in Fig.5.8, the total MS is 111 minutes and the total AET is 95 minutes. Thus, w.r.t. the RND rule, the MS is reduced

(improved) by 5.1% and the AET is augmented (worsened) by 2%. However, the aim of the PRI rule is not to reduce execution times but the minimization of the lateness of missions.

Notice that, being for phase 2, the PRI rule has been used in combination with the LUR rule for phase 1.

5.2.7 Discussion

From the simulation results presented in Fig.5.3–Fig.5.8 it can be noticed that the best AET (Fig.5.7) is reached with the rules MTPT and MD (AET = 79 minutes) which allow to save, on average, about 14 minutes w.r.t. the RND rule (AET = 93 minutes).

Instead, the best MS (Fig.5.5) is obtained with the rule MMT (MS = 108 minutes) which allows to save about 9 minutes w.r.t. the RND rule (MS = 117 minutes).

However, by considering each batch of missions individually, it can be observed that:

- The best MS for the batch 1 is obtained with the rule MMT;
- The best MS for the batch 2 is obtained with the rule MTPT;
- The best MS for the batch 3 is obtained with the rule PRI;
- The best MS for the batch 4 is obtained with the rule MD;

and that:

- The best AET for the batch 1 is obtained with the rule MD;
- The best AET for the batch 2 is obtained with the rule MTPT;
- The best AET for the batch 3 is obtained with the rule MTPT;
- The best AET for the batch 4 is obtained with the rule MTPT;

Thus, by testing via simulation every dispatching rule in advance and by choosing for each batch of missions the best one, better optimization results could be obtained. This is what is discussed in the following paragraph.

5.3 Variable-Rule Simulations: Reaching an Objective

As already highlighted, by fully exploiting the operational control scheme of Fig. 4.5 and letting the dispatching rule vary, better values of AET and MS, can be obtained. Indeed, each rule's behavior strictly depends on the input data of the control problem.

In this paragraph the simulation results obtained by fixing an objective function, and letting the dispatching rule vary, are presented on the same case study of Paragraph 5.2. Thus, a set of 6 resources and the same 4 batches of 60 missions have been considered.

5.3.1 Makespan Minimization

The minimization of the MS is here assumed as objective function. Thus, according to the notation of Paragraph 4.2 ($x_1 = MS$ $x_2 = AET$ and $x_3 = NLM$) the objective consists in minimizing

$$z = c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3$$

with

$$c_2 = 0$$

and

$$c_3 = 0.$$

As shown in Fig. 5.9, if the objective is to minimize the makespan, the best rules to choose for each batch of missions are in order: MMT, MTPT, PRI, MD.

With these choices the MS reaches the value of 96 minutes which is lower than the best value obtained with a fixed dispatching rule approach (108 minutes of the MMT rule). Thus, w.r.t. the

Objective	Batch	Rule Chosen	RND	MD	MMT	MPT	MTPT	PRI
			Makespan (sec)					
Minimize MS	1	MMT	1588	1467	1442	1691	1615	1648
	2	MTPT	1437	1441	1422	1588	1387	1449
	3	PRI	2383	2138	2102	2537	2493	1661
	4	MD	1567	1285	1299	1389	1299	1693
	TOT AET (sec)		5089 sec			85 min		
	TOT MS (sec)		**5775 sec**			**96 min**		

Figure 5.9 Variable rule simulation results: Minimization of the makespan.

RND rule, the MS is reduced (improved) by 6.8% and the AET is reduced (improved) by 18% (3% more than with the MMT rule).

5.3.2 Average Ending Time Minimization

The minimization of the AET is here assumed as objective function. Thus, according to the notation of Paragraph 4.2 ($x_1 = MS$ $x_2 = AET$ and $x_3 = NLM$) the objective consists in minimizing

$$z = c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3$$

with

$$c_1 = 0$$

and

$$c_3 = 0.$$

As shown in Fig. 5.10, if the objective is to minimize the average ending time, the best rules to choose for each batch of missions are in order: MD, MTPT, MTPT, MTPT.

With these choices the AET reaches the value of 78 minutes which is lower than the best value obtained with a fixed dispatching rule approach (79 minutes of the MTPT rule). Thus, w.r.t. the RND rule, the MS is reduced (improved) by 6.8% and the AET is reduced (improved) by 16.1%.

Objective	Batch	Rule Chosen	RND	MD	MMT	MPT	MTPT	PRI
			Average Ending Time (sec)					
Minimize AET	1	MD	1442	1279	1283	1570	1309	1516
	2	MTPT	1310	1173	1204	1347	1152	1304
	3	MTPT	1418	1165	1138	1385	1138	1443
	4	MTPT	1420	1149	1158	1252	1134	1410
	TOT AET		**4703 sec**			**78 min**		
	TOT MS		6375 sec			106 min		

Figure 5.10 Variable rule simulation results: Minimization of the average ending time.

5.4 Variable-Rule Simulations: Online Control

In this paragraph, the two reaction strategies described in Paragraph 4.4.2 are compared in non-nominal conditions. Simulations are run on a case study consisting of batches of 50 missions, in which 6 resources have been considered. A due date (and priority) has been introduced which is the same for all the missions of the same batch.

In nominal conditions, 3 batches are supposed to be available at certain time instants as shown in Fig. 5.11a. Four non-nominal situations have been considered:

1. The batch b_2 is missing (Fig. 5.11);
2. The batch b_2 is delayed (Fig. 5.13);
3. An unforseen urgent batch (b_4) arrives at a certain time instant (Fig. 5.15);
4. The resource R_2 brakes (Fig. 5.17).

For the simulations, a multi-objective objective function has been used. Thus, according to the notation of Paragraph 4.2

($x_1 = MS$ $x_2 = AET$ and $x_3 = NLM$) the objective consists in minimizing

$$z = c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3.$$

The weights c_1 c_2 and c_3 have been tuned so as to reach a good compromise between the minimization of the makespan, of the average ending time and the number of late missions, i.e.

$$c_1 = 0.4,$$

$$c_2 = 0.1$$

and

$$c_3 = 0.5.$$

This means that the minimization of the makespan is not the only objective to be reached. Thus, worse time optimization results, than the ones previously obtained, are expected with the advantage that also the minimization of the number of late missions is pursued.

5.4.1 Nominal Situation and Missed Missions

In Fig. 5.11a the nominal behavior of the system is depicted. As shown in Fig. 5.12 the nominal MS is 99 minutes and for only 4 missions the due date cannot be respected.

Notice that it has been supposed that the execution of a new batch of missions cannot be started until resources are all idle.

As depicted in Fig. 5.11b, with the reaction strategy 1 when the batch b_2 is missing, the batch b_3 (which is assumed available) is executed soon after the batch b_1. The simulation results are reported in Fig. 5.12. It is obvious that the entire batch b_2 is never executed since it is unavailable. Thus, all the 50 missions of the batch b_2 result late.

Notice that the time duration of the batch b_3 varies depending on if b_3 is executed after b_1 or b_2. This depends on the initial execution conditions of b_3 which coincide with the final conditions of the batch b_1 or of the batch b_2 respectively. For example, the

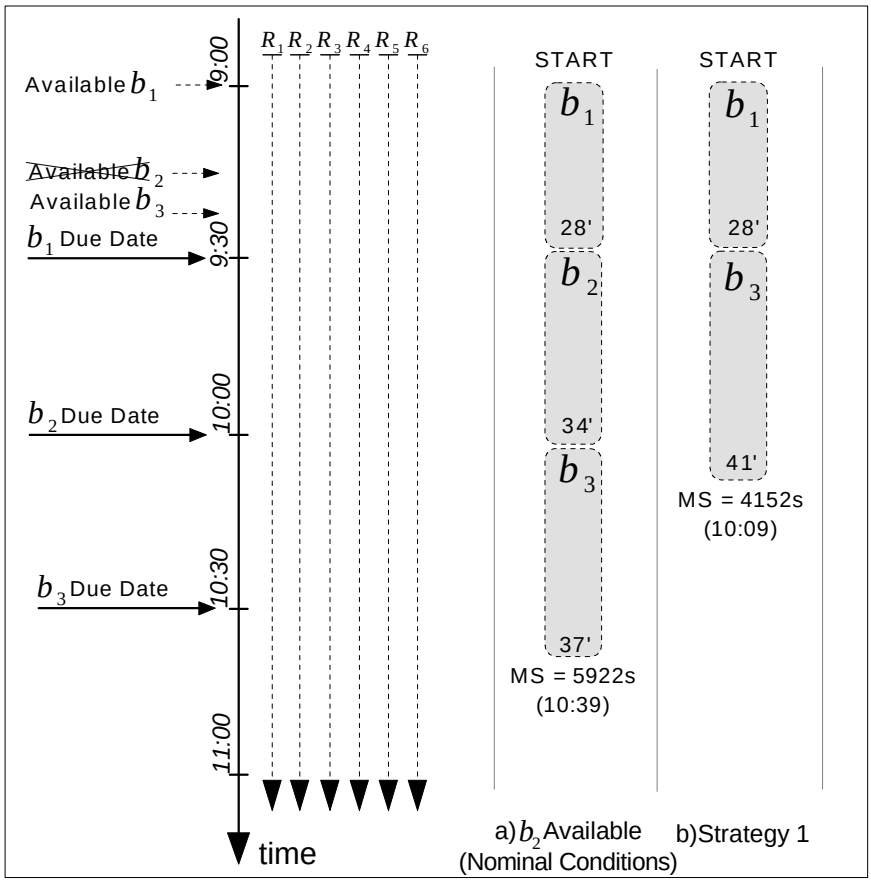

Figure 5.11 First non-nominal situation (Batch b_2 Missed) **a)** Nominal conditions **b)** Reaction strategy 1.

		Nominal		Strategy 1	
Batch b2 Missed	**Late Missions**	batch1: 0 batch2: 1 batch3: 3		batch1: 0 batch2: 50 batch3: 0	
	NLM	4		50	
	MS	5922 sec	99 min	4152 sec	69 min
	Rescheduling Time	/	/	/	/

Figure 5.12 Simulation results for missed missions (batch b_2): Nominal conditions and reaction strategy 1.

final position of resources after the execution of a batch strongly influences the future time duration of the next batch of missions.

The reaction strategy 2 does not make sense in this situation, since b_2 is supposed unavailable and not delayed.

5.4.2 Delayed Missions

In Fig. 5.13 the reaction strategy 1 and 2, when the batch b_2 is delayed, are depicted.

As shown in Fig. 5.14 with the reaction strategy 1 (Fig. 5.13a) a MS of 95 minutes is obtained and 50 missions are late. Since the batch b_2 is not available in time, the batch b_3 (which is assumed available) is executed soon after the batch b_1.

It is important to underline that the makespan in this case is different from the one obtained in the nominal situation (99 minutes) just because, as observed before, a different execution order of the batches (b_1, b_2, b_3 in the nominal case, b_1, b_3, b_2 when b_2 is delayed) modifies the initial conditions and consequently the execution time, of each batch.

With the reaction strategy 2 (Fig. 5.13b), which corresponds to an online kind of control, a MS of 75 minutes is obtained and 21 missions are late. Thus, by completely rescheduling the missions when b_2 finally becomes available, 20 minutes are gained w.r.t.

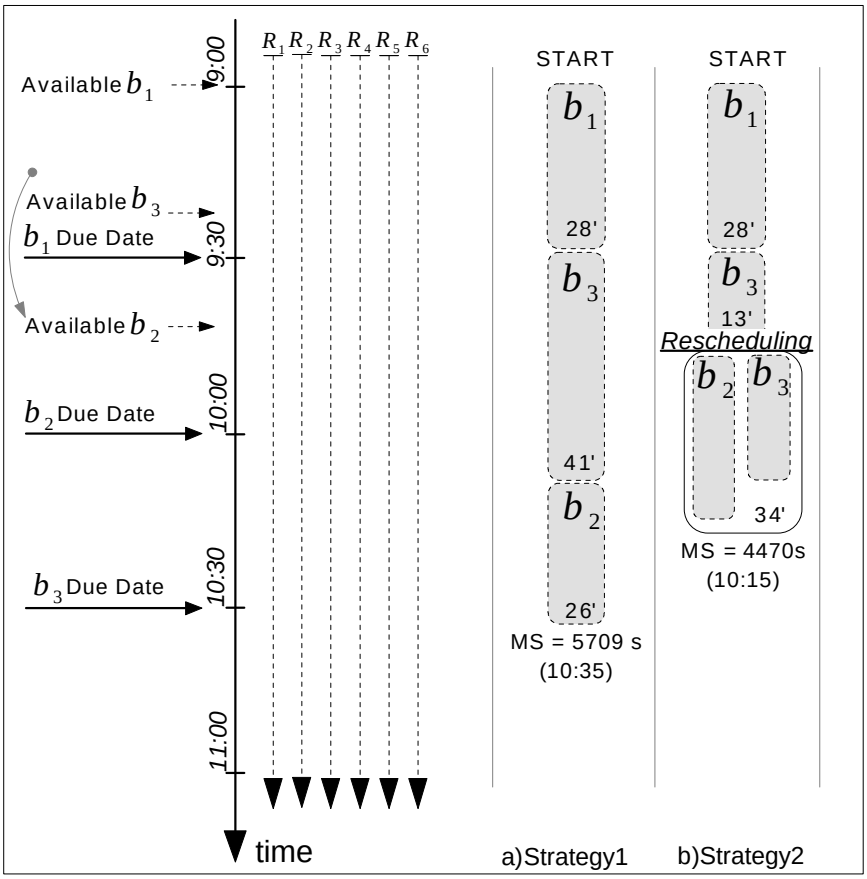

Figure 5.13 Second non-nominal situation (Batch b_2 Delayed) **a)** Reaction strategy 1 **b)** Reaction strategy 2.

		Strategy 1		Strategy 2	
Batch b2 Delayed	**Late Missions**	batch1: 0 batch2: 50 batch3: 0		batch1: 0 batch2: 21 batch3: 0	
	NLM	50		21	
	MS	5709 sec	95 min	4470 sec	75 min
	Rescheduling Time	/	/	74 sec	1,2 min

Figure 5.14 Simulation results for delayed missions (batch b_2): Reaction strategy 1 and reaction strategy 2.

strategy 1 (MS improved by 21%) and the number of late missions is reduced of 29 missions.

In the reaction strategy 2 the reschedule is accomplished in $74s$ (about 1.2 minutes). This rescheduling time is negligible with respect to the time horizons of the considered batches. Notice that when b_2 finally becomes available the execution of b_3 is stopped and a new model including b_2 is computed.

5.4.3 Urgent Missions Arrival

In Fig. 5.15 the reaction strategy 1 and 2, when suddenly arrives the urgent batch b_4, are depicted.

As shown in Fig. 5.16 with the reaction strategy 1 (Fig. 5.15a) a MS of 124 minutes is obtained and 69 missions are late. Since the batch b_4 is urgent, in the reaction strategy 1 b_4 is executed soon after the batch b_2, before the batch b_3.

With the reaction strategy 2 (Fig. 5.15b), which corresponds to an online kind of control, a MS of 111 minutes is obtained and only 19 missions are late. By completely rescheduling the missions when b_4 arrives, 12 minutes are gained w.r.t. strategy 1 (MS improved by 10.5%) and the number of late missions is reduced of 50 missions.

In the reaction strategy 2 the reschedule is accomplished in

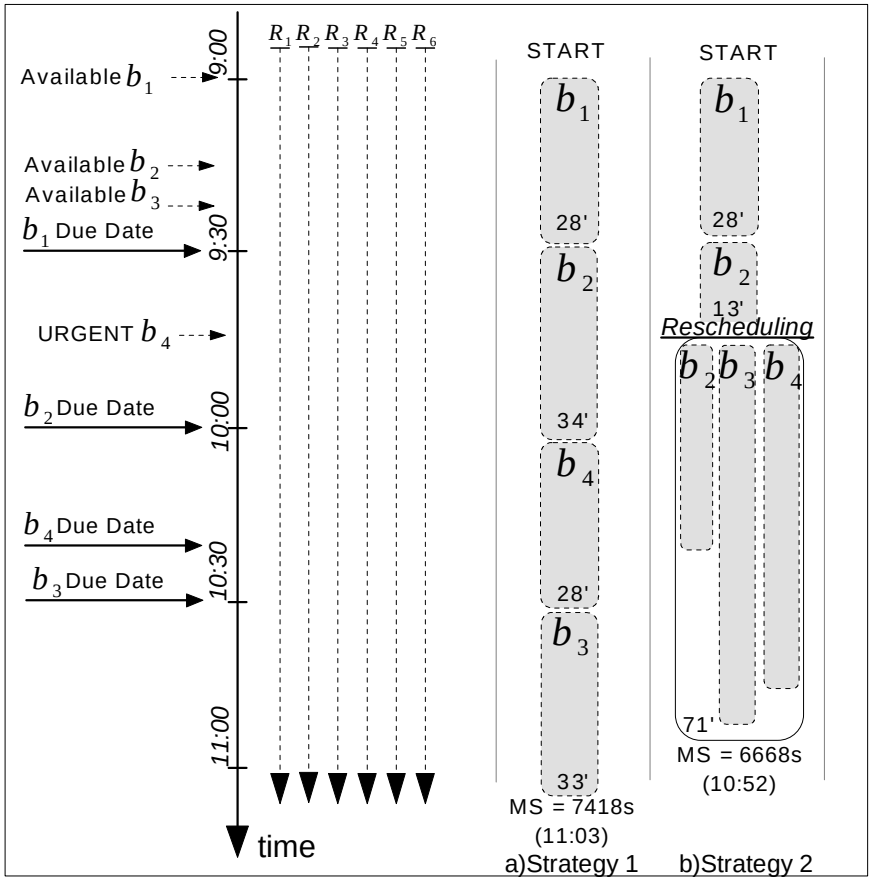

Figure 5.15 Third non-nominal situation (Unforseen arrival of the batch b_4) **a)** Reaction strategy 1 **b)** Reaction strategy 2.

		Strategy 1		Strategy 2	
Batch b4 Urgent	**Late Missions**	batch1: 0 batch2: 1 batch3: 50 batch4: 18		batch1: 0 batch2: 5 batch3: 8 batch4: 6	
	NLM	69		19	
	MS	7418 sec	124 min	6668 sec	111 min
	Rescheduling Time	/	/	244 sec	4 min

Figure 5.16 Simulation results for urgent missions arrival (batch b_4): Reaction strategy 1 and reaction strategy 2.

244s (about 4 minutes). This rescheduling time can be considered negligible with respect to the time horizons of the considered batches. Notice that when b_4 arrives the execution of b_2 is stopped and a new model including b_4 is computed.

5.4.4 Breakdown of a Resource

In Fig. 5.17 the reaction strategy 1 and 2, when the resource R_2 breaks, are depicted.

As shown in Fig. 5.18 with the reaction strategy 1 (Fig. 5.17a) a MS of 119 minutes is obtained and 15 missions are late. Consider that, with the reaction strategy 1, when R_2 breaks the missions of b_2 and b_3 associated to R_2 are momentarily suspended. Then they are assigned to the remaining resources when they have completed their joblists and have become available. The makespan is greater than the one obtained in nominal conditions since, from the moment that R2 breaks, missions are executed by five resources instead of six resources.

With the reaction strategy 2 (Fig. 5.17b), which corresponds to an online kind of control, a MS of 85 minutes is obtained and only 4 missions are late. By completely rescheduling the missions when the resource R_2 breaks, 34 minutes are gained w.r.t. strategy 1 (MS improved by 28.6%) and the number of late missions is

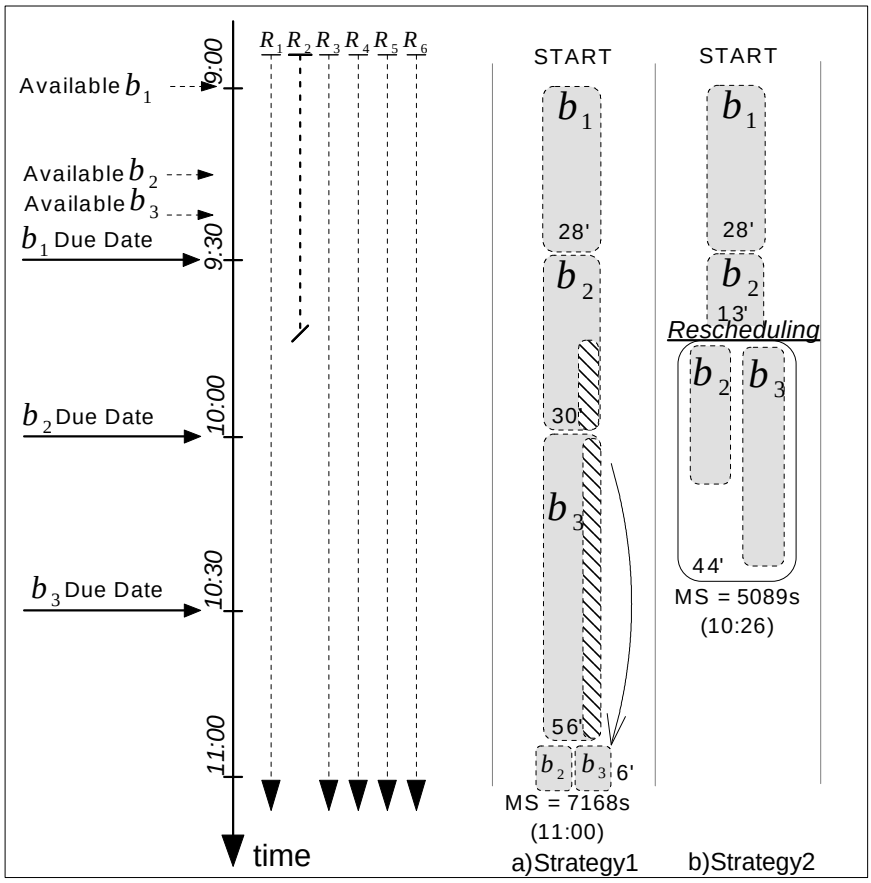

Figure 5.17 Fourth non-nominal situation (Breakdown of the resource R_2) **a)** Reaction strategy 1 **b)** Reaction strategy 2.

		Strategy 1		Strategy 2	
Resource R2 Breaks	**Late Missions**	$batch_1$: 0 batch2: 1 batch3: 14		$batch_1$: 0 batch2: 4 batch3: 0	
	NLM	15		4	
	MS	7168 sec	119 min	5089 sec	85 min
	Rescheduling Time	/	/	72 sec	1,2 min

Figure 5.18 Simulation results for breakdown of the resource R_2: Reaction strategy 1 and reaction strategy 2.

reduced of 11 missions.

In the reaction strategy 2 the reschedule is accomplished in $72s$ (about 1.2 minutes). This rescheduling time is negligible with respect to the time horizons of the considered batches. Notice that when the resource R_2 breaks the execution of b_2 is stopped and a new model which does not include R_2 is computed.

Chapter 6

Conclusions and Future Research

In this chapter the main conclusions of the thesis are summarized. Some possible future developments for this research are also discussed.

6.1 What has been accomplished

Material Handling Systems (MHSs) are systems where several kinds of resources like automated vehicles, manned vehicles and on-foot storemen have to execute a list of moving orders. They are everywhere in production plants, assembly lines, product distribution, logistics, etc.. It has been estimated that, although not adding value in the manufacturing process, Material Handling (MH) usually influences great part of a company's operation costs, especially, for example, in the food chain.

Optimizing MH activities means having shorter response times and an increased throughput of the plant. The importance of this optimization process is very high in today's companies where MHSs are usually very complex. The term "complex" is here used to denote MHSs which are very turbulent and heterogeneous, i.e. MHS in which the mission set and the resource set are variable, and items and resources are strongly heterogeneous. These com-

plexities influence both modeling and control aspects. Indeed, enough general and flexible models are hard to be found in the literature of MHSs which is usually addressed only to automated warehousing systems. From the control point of view instead, the strong combinatorial nature of control problems in MHSs usually makes the design of a control law very tough. Closed form analytical solutions are almost impossible to be designed and simulation is more and more fundamental to evaluate the effects of a control action which cannot be analytically predicted.

In this thesis a unique architecture for the modeling and the control of complex MHSs has been proposed.

The (two-level) modeling framework incorporates a formal model, based on Colored Timed Petri nets (CTPNs), and an high level model. CTPNs have proved to be a very effective tool for both the modeling and the simulation of discrete event systems in general. In this work they are used to model, independently from the specific context, the activities in a MHS leading to a model structure that is the same for every MHS. Moreover, they are used to easily detect conflicts and confusions in the system. The CTPN model is extended with an high level model called "Auxiliary Process" which adds a further modeling level providing information specific for each plant and constraints on the execution of missions which are not convenient to include in the formal model.

The control is realized through dispatching rules which are easily expressed and implemented thanks to the information made available by the two-level modeling framework. Through the simulation of the model it is always known when a control decision must be made and who is involved in it. Look-ahead simulations are used to test in advance several dispatching rules over a certain time horizon. This allows to dynamically vary the control action over time at a very low cost.

The developed control architecture has been validated through several simulations based on real case studies. Simulations have been done exploiting the PNs simulation tool PNetLab [BCC07].

Through simulation it has been confirmed that fixed-rule control approaches are outperformed by variable-rule ones. In fact, to

let the control action vary, in order to always choose the best one for each data set, produces better results than fixing the control action.

Online control issues have also been taken into account. Through simulations, the improvements of the performances introduced by moving from an offline to an online kind of control (which allows to react in real-time to disturbances), have been quantified. Two kinds of disturbances have been considered:

1. Disturbances on the mission set:
    - *Missed Missions Arrival*;
    - *Delayed Missions Arrival*;
    - *Unexpected Urgent Missions Arrival*;
2. Disturbances on the resource set:
    - *Breakdown of a Resource*;
    - *Adding of a Resource.*

The proposed architecture is ready to be used for the online control of real MHSs in which measures from the plant are available. Several new technologies, which are available today at a very low cost, may help to this purpose.

6.2 Future research directions

In the control action $u_k = \mu_k(q_k)$ described in Chapter 4, the index k identifies one dispatching rule among a set of rules, which optimize a certain index. Several dispatching rules have been proposed and, assumed as objective the minimization of the execution time of missions, some of them have been included in the test set when presenting simulation results.

However, the choice of one (the best) rule among a set, and its use to solve the control problem, does not produce a global optimum solution which could only be found by searching among

all the possible dispatching combinations. The problem is that dispatching rules are applied locally to each conflict and only by searching in a tree containing all the dispatching decisions over time, the global optimum solution could be found.

Of course sophisticated search algorithms could be used to minimize the search time along the tree, but the highly combinatorial nature of the control problem makes the dimensions of such tree usually very large. Thus, particularly when simulation is needed to determine the arc weights of the tree (this is the case of time minimization objectives), the population of the tree could result in an impossible goal, especially at runtime.

Dispatching rules represent a way to overcome this problem since, in the reaching of a certain objective, they allow to evaluate specific search direction rather than all possible combinations. New dispatching rules can be always thought and, obviously, more rules are simulated and compared, much closer to the global optimum the obtained control solution is. However, for online control purposes, and to maintain computation times limited, only a subset of rules should be considered.

Thus, in this optics the main problem to solve is the choice of a subset of dispatching rules, among a set of available ones, to be used in order to obtain a control solution as much closer to the optimum as possible. What it is not yet clear is the bind among each dispatching rule and the objective functions. If a relationship exists, of course this function is not analytically expressible. This is due to the fact that the behaviour of the rules is local and data-dependent. To know how every dispatching rule influences an objective function, i.e. to express the objective function as a function of the rules and not of the conflicts to be solved, would allow to compute the subset of (best) rules to be used for the online control.

One major future activity will be to investigate just the use of off-line simulations to solve this problem. I.e., instead of vainly try to analytically describe the relationship among dispatching rules and objective functions, the idea is to use off-line simulations to determine, for each data set, the subset of best rules to be

compared through look-ahead online simulations.

Another objective to be reached, in a short term future, is the shortening of simulation time. This could be obtained in several ways.

At first, multi-core architectures of modern processors could be better exploited. Indeed, look-ahead simulations under different dispatching rules could be run in parallel rather than sequentially. This would drastically reduce decision times, and costs would not be increased since multi-core architectures are nowadays available everywhere. For instance, it has been estimated that four parallel simulations (for four dispatching rules) on a quad-core processor last almost the same time than one simulation on a single-core processor. Thus, since simulations on single-core processors are run sequentially, four dispatching rules, for example, could be simulated in nearly one fourth of the time on a quad-core processor.

Finally, the adoption of hybrid and continuous Petri nets models to fight against the state explosion issue of the control problem could be investigated. One way to face the combinatorial explosion problem is to use some kind of relaxation. The so-called "fluidification" [SR04] of Petri nets, for example, is a relaxation technique which is frequently used to deal with very populated or high traffic systems. Through the reduction of the control problem dimensions a shortening of simulation times would be implicitly obtained.

Bibliography

[ABCC05] F. Amato, F. Basile, C. Carbone, and P. Chiacchio, *An approach to control automated warehouse systems*, Control Engineering Practice **13** (2005), 1223–1241.

[App77] J.M. Apple, *Plant layout and material handling, 3rd ed.*, John Wiley & Sons Inc, 1977.

[BCC07] F. Basile, C. Carbone, and P. Chiacchio, *Simulation and analysis of discrete-event control systems based on petri nets using pnetlab*, Control Engineering Practice **15** (2007), 241–259.

[BCD08a] F. Basile, P. Chiacchio, and D. Del Grosso, *Modelling automation systems by UML and Petri nets*, Proceedings 9th International Workshop on Discrete Event Systems (WODES 08), Gotheborg, Sweden, 28-30 May (2008), 308–313.

[BCD08b] ———, *UML-based modeling and model-driven development of distributed control systems*, Proceedings 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 08), Hamburg, Germany, 15-18 Sept (2008), 1120–1127.

[BCD09a] ———, *An approach to control generalized warehouses*, Proceedings 14h IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 09), Mallorca, Spain, 22-26 Sept (2009), 1–8.

[BCD09b] ———, *A control oriented model of generalized warehouses based on colored timed Petri nets*, Proceedings IEEE International Conference on Automation Science and Engineering (CASE 09), Bangalore, India, 22-25 August (2009), 48–53.

[BCD09c] ———, *A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri net*, Computer Standards and Interfaces (Industrial Networking Standards for Real-time Automation and Control) **31** (2009), no. 3, 528–538.

[BCDD06] F. Basile, P. Chiacchio, G. De Tommasi, and D. Del Grosso, *Applications of RFID technology in the warehouses management*, International Congress on Methodologies for Emerging Technologies in Automation (ANIPLA 2006), Rome, Italy, 13-15 Nov (2006).

[BCDD08] ———, *Performing fault diagnosis for PNs using G-Markings: A benchmark case*, Proceedings 9th International Workshop on Discrete Event Systems (WODES 08), Gotheborg, Sweden, 28-30 May (2008), 137–143.

[BLK+02] S. Bogdan, F.L. Lewis, Z. Kovacic, A. Gurel, and M. Stajdohar, *An implementation of the matrix-based supervisory controller of flexible manufacturing systems*, IEEE Transactions on Control Systems Technology **10** (2002), no. 5, 709–716.

[BLKM06] S. Bogdan, F.L. Lewis, Z. Kovacic, and J.J. Mireles, *Manufacturing systems control design*, Springer, 2006.

[CL08] C.G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, Springer, 2008.

[Cra97] Y. Crama, *Combinatorial optimization models for production scheduling in automated manufacturing*

systems, European Journal of Operational Research **99** (1997), no. 1, 136–153.

[CT09] J. Chung and J.M.A. Tanchoco, *Material handling automation in production and warehouse systems*, Springer Handbook of Automation **F** (2009), 961–979.

[CU92] L.K. Church and R. Uzsoy, *Analysis of periodic and event-driven rescheduling policies in dynamic shops*, International Journal of Computer Integrated Manufacturing **5** (1992), no. 3, 153–163.

[CVP99] R. Champagnat, R. Valette, and H. Pingaud, *Formal methods for batch production systems*, Formal Methods and Manufacturing (F.M. & M. '99), Prensas Universitarias di Zaragoza (1999), 1–20.

[DF04] M. Dotoli and M.P. Fanti, *Coloured timed petri net model for real-time control of automated guided vehicle systems*, Int. Journal of Production Research **42** (2004), no. 9, 1787–1814.

[GZN+06a] V. Giordano, J.B. Zhang, D. Naso, F. Lewis, A. Carbotti, and N.T. Jye, *A matrix-based framework for combined supervisory and operational control of an industrial warehouse*, IEEE International Conference on Industrial Informatics, Singapore (2006), 201–206.

[GZN+06b] V. Giordano, J.B. Zhang, D. Naso, M.M. Wong, F.L. Lewis, and A. Carbotti, *Matrix-based discrete event control of automated material handling systems*, Proceedings of the 45th IEEE Conference on Decision & Control (2006), 1–6.

[GZNL08] V. Giordano, J.B. Zhang, D. Naso, and F. Lewis, *Integrated supervisory and operational control of a warehouse with a matrix-based approach*, IEEE Transactions on Automation Science and Engineering **5** (2008), no. 1, 53–70.

[Har95] C.M. Harmonosky, *Simulation-based real-time scheduling: Review of recent developments*, Proceedings of the 1995 Winter Simulation Conference (1995), 220–225.

[HE09] S.S. Heragu and B. Ekren, *Materials handling system design*, Environmentally Conscious Materials Handling (Myer Kutz, ed.), John Wiley & Sons, Inc., 2009, pp. 1–29.

[HL06] Y.C. Ho and H.C. Liu, *A simulation study on the performance of pickup-dispatching rules for multiple-load agvs*, Computers & Industrial Engineering **51** (2006), 445–463.

[HR91] C.M. Harmonosky and S.F. Robohn, *Real-time scheduling in computer integrated manufacturing: a review of recent research*, International Journal of Computer Integrated Manufacturing **4** (1991), no. 6, 331–340.

[JK98] K.C. Jeong and Y.D. Kim, *A real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules*, International Journal of Production Research **36** (1998), no. 9, 2609–2626.

[JM98] A.S. Jain and S. Meeran, *A state-of-the-art review of job-shop scheduling techniques*, 1998.

[JM07] J. Jokinen and J.L. Martinez Lastra, *Agent-based control of rapidly reconfigurable material handling system*, IEEE Conference on Emerging Technologies and Factory Automation (2007), 1217–1224.

[JR98] A. Jones and L.C. Rabelo, *Survey of job shop scheduling techniques*, Available in the World Wide Web at http://www.nist.gov/mel/, 1998.

[KD91] I. Koh and F. Di Cesare, *Synthesis rules for colored petri nets and their applications to automated manufacturing systems*, Proceedings of the 1991 IEEE International Symposium on Intelligent Control, 13-15 August 1991, Arlington, Virginia (1991), 152–157.

[KK94] M.H. Kim and Y.D. Kim, *Simulation-based real-time scheduling in a flexible manufacturing system*, Journal of Manufacturing Systems **13** (1994), no. 2, 85–93.

[Kow99] S. Kowalewsky, *Formal methods and the processing industries: status and prospects from ad academic perspective*, Formal Methods and Manufacturing (F.M. & M. '99), Prensas Universitarias di Zaragoza (1999), 75–91.

[LDM09] J. Li, X. Dai, and Z. Meng, *Automatic reconfiguration of petri net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach*, IEEE Transactions on Automation Science and Engineering **6** (2009), no. 1, 156–167.

[LS96] H.F. Lee and S.K. Schaefer, *Retrieval sequencing for unit-load automated storage and retrieval systems with multiple openings*, Int. J. of Production Research **34** (1996), no. 10, 2943–2962.

[Mal98] C.J. Malmborg, *Analysis of storage assignment policies in less than unit load warehousing systems*, International Journal of Production Research **36** (1998), no. 12, 3459–3475.

[ML01] J.J. Mireles and F.L. Lewis, *Intelligent material handling: Development and implementation of a matrix-based discrete-event controller*, IEEE Transactions on Industrial Electronics **48** (2001), no. 6, 1087–1097.

[Mur89] T. Murata, *Petri nets: Properties, analysis and applications*, Proceedings of IEEE **77** (1989), no. 4, 541–580.

[NST07] D. Naso, M. Surico, and B. Turchiano, *Reactive scheduling of a distributed network for the supply of perishable products*, IEEE Transactions on Automation Science and Engineering **4** (2007), no. 3, 407–423.

[PDL99] L. Pietrac, B. Denis, and J-J. Le Sage, *An approach of formal meta-modeling for the design methods of automated manufacturing systems*, Formal Methods and Manufacturing (F.M. & M. '99), Prensas Universitarias di Zaragoza (1999), 65–74.

[Pin05] M.L. Pinedo, *Planning and scheduling in manufacturing and services*, Springer, 2005.

[PJ05] A. Polic and K. Jezernik, *Closed-loop matrix based model of discrete event systems for machine logic control design*, IEEE Transactions on Industrial Informatics **1** (2005), no. 1, 39–46.

[Pro07] J.M. Proth, *Scheduling: New trends in industrial environment*, Annual Reviews in Control **31** (2007), no. 1, 157–166.

[Rev05] S.A. Reveliotis, *Real-time management of resources allocation systems - a discrete event systems approach*, Springer's International Series, 2005.

[RVL92] S. Ramaswamy, K.P. Valavanis, and S.P. Landry, *Modeling, analysis and simulation of a materials handling system with extended petri nets*, Proceedings of the 31st IEEE Conference on Decision and Control (1992), 1665–1672.

[Smi95] S.F. Smith, *Reactive scheduling systems*, Kluwer Publishing, 1995.

[SR04] M. Silva and L. Recalde, *On fluidification of petri nets: from discrete to hybrid and continuous models*, Annual review in control **28** (2004), 253–266.

[SRW99] Y.J. Son, H. Rodriguez-Rivera, and R.A. Wysk, *A multi-pass simulation-based, real-time scheduling and shop floor control system*, Transactions of the Society for Computer Simulation **16** (1999), no. 4, 159–172.

[SWJ03] Y.J. Son, R.A. Wysk, and A.T. Jones, *Simulation-based shop floor control: Formal model, model generation and control interface*, IEE Transactions **35** (2003), 29–48.

[TL97] D.A. Tacconi and F.L. Lewis, *A new matrix model for discrete event systems: application to simulation*, IEEE Control Systems Magazine **17** (1997), 62–71.

[Van99] J.P. Van den Berg, *A literature survey on planning and control of warehousing systems*, IIE Transactions **31** (1999), 1–13.

[Van00] R. Van Der Meer, *Operational control of internal transport*, Ph.D. thesis, Delft University, 2000.

[Vis02] I.F.A. Vis, *Planning and control concepts for material handling systems*, Erim Ph.D. Series Research in Management 14, 2002.

[VRL94] K.P. Valavanis, S. Ramaswamy, and S.P. Landry, *Extended petri net-based modeling, analysis and simulation of an intelligent materials handling system*, Journal of Intelligent Robotic Systems **10** (1994), 79–108.

[WTZ+07] M.M. Wong, C.H. Tan, J.B. Zhang, L.Q. Zhuang, Y.Z. Zhao, and M. Luo, *On-line reconfiguration to enhance the routing flexibility of complex automated material handling operations*, Robotics and Computer-Integrated Manufacturing **23** (2007), 294–304.

[WW89] S.Y.D. Wu and R.A. Wysk, *An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing*, International Journal of Production Research **27** (1989), no. 9, 1603–1623.

[YN85] M. Yamamoto and S.Y. Nof, *Scheduling/rescheduling in the manufacturing operating system environment*, International Journal of Production Research **26** (1985), no. 4, 705–722.

[ZFY05] W. Zhang, T. Freiheit, and H. Yang, *Dynamic scheduling in flexible assembly system based on timed petri nets model*, Robotics and Computer-Integrated Manufacturing **21** (2005), no. 6, 550–558.