Simulation-based optimization of discrete event systems with alternative structural configurations using distributed computation and the Petri net paradigm

Juan-Ignacio Latorre and Emilio Jiménez

SIMULATION 2013 89: 1310 originally published online 31 October 2013
DOI: 10.1177/0037549713505761

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
89(11) 1310–1334

DOI: 10.1177/0037549713505761
sim.sagepub.com
SAGE SCS

Simulation-based optimization of discrete event systems with alternative structural configurations using distributed computation and the Petri net paradigm

Juan-Ignacio Latorre[1] and Emilio Jiménez[2]

Abstract

Decision-making on discrete event systems with alternative structural configurations is a field with application to the efficient design and operation of many systems, ranging from manufacturing facilities to communication networks. The solution of this problem may be afforded by its transformation into an optimization problem. A variety of statements for this optimization problem can be presented by using different formalisms able to describe the model of the system. These different statements allow developing diverse optimization algorithms for solving the problem, which may be very demanding for a computer. In this paper, several approaches are presented in order to reduce the computing requirements needed by the mentioned algorithms, some of them are implemented in one processor and others are based on distributed computing. In particular, this paper presents a new distributed methodology, which associates sets of alternative structural configurations of the system to different alternative aggregation Petri net (AAPNs), regarding the number of available processors. Under certain conditions, this methodology alleviates the computational requirements for every processor and speeds up the optimization process. A case-study is presented and different techniques are applied to solve it, for illustrating diverse distributed and non-distributed methodologies, regarding the available processors, as well as for comparing their relative performance.

Keywords

Distributed computation, Petri nets, alternative aggregation Petri nets, AAPN, distributed optimization, decision-making, discrete event systems

1. Introduction

1.1. Discrete event systems and the Petri nets

A discrete event system (DES) is characterized by a discrete state-space and by a state transition mechanism, which is event-driven.[1] A DES can be found in many technological fields with a significant presence of computers, such as communication networks, supply chains, and manufacturing processes.

In the process of the design and operation of a DES, usually some degrees of freedom exist, associated to a set of alternative feasible configurations. These degrees of freedom lead to the concept of an undefined DES, which is converted into a defined DES by choosing a unique feasible configuration for the degrees of freedom. An intuitive insight of this idea can be seen in Figure 1. The choice of a given configuration for the degrees of freedom can be afforded by stating and solving a decision-making process, devoted to specifying the best alternative configuration for this purpose.

In order to develop repeatable, precise, and efficient methodologies for solving the mentioned type of decision problems, it is convenient to develop a formal model for the undefined DES. Petri nets consist of a family of formalisms, which can cope with parallelism and concurrence,

[1]Department of Mechanical Engineering Energetics and Materials, Public University of Navarre, Spain
[2]Department of Electrical Engineering, University of La Rioja, Spain

Corresponding author:
Juan-Ignacio Latorre, Department of Mechanical Engineering Energetics and Materials, Public University of Navarre, Campus of Tudela. Ctra. Tarazona s/n. 31500 Tudela, Spain.
Email: juanignacio.latorre@unavarra.es

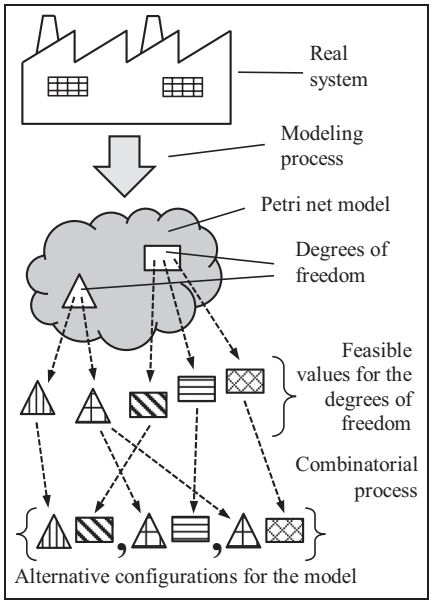

Figure 1. Undefined Petri net.

characteristic behaviors of complex systems. Petri nets show a simple and intuitive graphic representation, a useful matrix-based description of the model, and are equipped with a broad range of theoretical results, available for their analysis.[2] The formalization of the rest of the elements of the statement of the decision problem leads to a constraint satisfaction problem or, in a more general statement, to an optimization problem.

The optimization problem based on a real DES is commonly related to a combinatorial explosion, when considering the size of the solution space;[3,4] hence, performing an exhaustive exploration requires unaffordable computer resources. This situation leads to the development of efficient but approximated techniques to solve the problems.[5] An important group of these techniques include metaheuristics, a family of probabilistic search strategies.[3,4]

1.2. Classic approaches for the distributed simulation of Petri net models

Distributed computing is an approach that contributes to implementing fast algorithms for solving optimization problems, where simulation is afforded for evaluating the quality of the feasible solutions. In fact, the main motivation behind distributed simulation is to gain speed over traditional sequential simulation.

Distributed simulation has been applied in a large number of fields, comprising both discrete and continuous simulation, such as formation of stars and galaxies, design of global logistics systems, analysis and optimization of combustion in engines and furnaces, diagnosis in human organs, analysis of seismic responses on nuclear power plants, subsurface hydrology, or epidemiology. Research has been done on the topic of comparing conventional and distributed simulation for a given application in order to determine the conditions that allow one of these approaches to perform better than the others. See, for example, the work of Mustafee et al.[6]

Diverse successful approaches for distributed simulation have been developed, such as parallel DES simulation or grid-enabled DES simulation. These methodologies require synchronization techniques that can be conservative or optimistic, according to their policy regarding the local causality constraint. For example, Tang et al.[7] propose an optimistic parallel DES simulation approach including reverse computation techniques for recovering a previous state. Another optimistic parallel DES simulation is described by Vitali et al.,[8] that improves the use of memory caching architectures for ensuring correctness, communications, and event scheduling. A brief introduction on parallel and distributed simulation of a DES can be found in Ferscha.[9]

A computational grid can be seen as an ensemble of distributed and heterogeneous resources, which present enormous potential for running large-scale applications. By accessing grid computing using the appropriate middleware framework, complex simulations can experience a reduction in runtime.[10] Given a certain application, to be efficiently distributed on an existing computing infrastructure, a grid framework of this kind is required. For example, Andjelković et al.[11] describe the parallelization of the simulation of electronic circuits and systems, as well as the development of a grid interface. Furthermore, Evans et al.[12] has developed a platform that enables programming parallel and distributed DES simulations, either by parallelizing multiple executions of a given simulation or by the implementation of a distributed DES.

The complexity of scheduling and resource management in grid computing environments has led to the development of simulation tools for assessing the efficiency of different policies.[13] Other approaches profit from the capability of distributed systems for storing the data generated by a large series of complex simulations.[14] Web-based distributed simulation has also received attention from the researchers. Wainer et al.[15] report the use of a distributed simulation engine, which provides transparent sharing of computer power, DES specification (DEVS)-based models, data, and experiments in heterogeneous resources on a global scale.

A particular problem, which has received much attention from the research community, is the simulation of large and complex systems. For example, Timm and Pawlaszczyk[16] address the grid-based simulation of large-scale systems of autonomous decision-makers in the field of global logistics networks. Two problems in the distributed simulation of large-scale and detailed models, related to communication overhead and load-balancing, are

considered by D'Angelo et al.[17] in which the simulation runtime is eventually reduced. A synchronous conservative algorithm for distributed simulation is used by Martin[18] for optimizing the performance of detailed simulations of large mobile ad hoc networks. Packet-level simulations of large-scale computer networks performed on a variety of platforms and a large number of processors are reported by Fujimoto et al.[19] Theodoropoulos et al.[20] explores the challenges of large-scale and complex simulations profiting from the distributed resources over the Internet, managed by grid technologies, as well as explores collaborative model development.

A promising simulation methodology, which profits from a close relationship between the simulation system and the physical modeled system, for applications such as operational decision-making, is symbiotic simulation.[21]

Different methodologies have been developed for the distributed simulation of Petri nets. One of the most successful approaches is based on decomposing the Petri net model of the original DES into submodels. Each one of the Petri net submodels is assigned to a logical process that performs the simulation on a specific processor.

For example, Chiola and Ferscha[22] developed a heuristic for the decomposition of the Petri net model in a way that the concurrence and conflict resolution is internal to the processes. On the other hand, Hulaas[23] discussed the decomposition of the model by means of an object-oriented formal language, leading naturally to the description of distributed systems. The object-oriented techniques with the complement of autonomous agents allowed Holvoet and Verbaeten,[24] as well as Kuo[25] to model a Petri net as a set of autonomous cooperating entities. The problem of the time synchronization between processes and the global order of events is addressed by Nicol and Mao,[26] Knoke et al.,[27] and Zimmermann, Knoke and Hommel,[28] just to give a few examples.

Other approaches, which have been applied successfully for performing the distributed simulation of a DES, are described by Haggarty, Knottenbelt and Bradley,[29] as well as by Yoo, Cho and Yücesan.[30] For example, Zuberek[31] addresses the problem of the distributed generation of state-spaces for timed Petri nets. The performance of this process is enhanced by a reduction in the communication between processors by clustering appropriate states.

Regarding the field of simulation-based optimization of DESs modeled by Petri nets, different approaches have been presented and tested by the research community. In real applications, it is very common for the solution space to be large, hence not allowing an exhaustive exploration of every configuration of the Petri net by means of simulation. In order to reduce the runtime of an optimization, there are some approaches which apply a guided search that choose a small amount of solutions to explore. In some cases a manual selection of the solutions to be tested is the methodology applied.[32]

Nevertheless, metaheuristics are popular and usually efficient criteria for guiding the search in the solution space. For example, Mušič[33] addresses the Petri net-based job-shop scheduling by means of a combination of dispatching rules with a local search guided by a metaheuristic. On the other hand, Latorre et al.[4] describe the application of a genetic algorithm to solve a decision-making problem related to the design of a manufacturing facility. Zimmermann et al.[3] apply the technique of simulated annealing for solving a decision problem on the operation of a flexible manufacturing system.

Other approaches search for similarities in the reachable state-space of different solutions to prune the explored solution space.[34]

Bai et al.[35] describe an agent-based approach for supporting automated scheduling and planning. The chosen modeling formalism is the augmentation of colored Petri nets. Agents predict the probable future states of a system and the corresponding risks of reaching these states. The proposed methodology can enable agents to make accurate decisions in complex scheduling and planning problems.

The optimization of routing problems of automatic guided vehicles (GAV) in semiconductor manufacturing systems is addressed by Nishi and Maeno.[36] In order to reduce the size of the state-space, the Petri net is decomposed into several independent subnets, where the original shared places are removed and resource places are added. A partial solution is derived for each subnet and a penalty function algorithm integrates them to obtain a solution for the complete system.

1.3. New methodology proposed for the distributed simulation of Petri net models

The new methodology for distributed simulation, which is proposed in this paper, is also based on the decomposition of a Petri net model of a DES. Nevertheless, there are important differences with the standard approaches mentioned in the previous section.

Firstly, the proposed new methodology is applied to the Petri net model of a particular kind of system, which includes degrees of freedom associated to a set of alternative structural configurations. This kind of system belongs to the category of undefined DESs, previously mentioned in section 1.1, and is very common in the design process of DESs and usual in their operation. See, for example, Tsinarakis et al.[37], Zimmermann et al.,[3] or Zhou and Venkatesh.[32] An intuitive representation of an undefined DES and its model, an undefined Petri net, is shown in Figure 1.

The degrees of freedom in the Petri net model have been depicted as missing pieces of information that can be filled from a set of feasible values. In Figure 1, the model of the system is depicted as a cloud; the missing pieces of

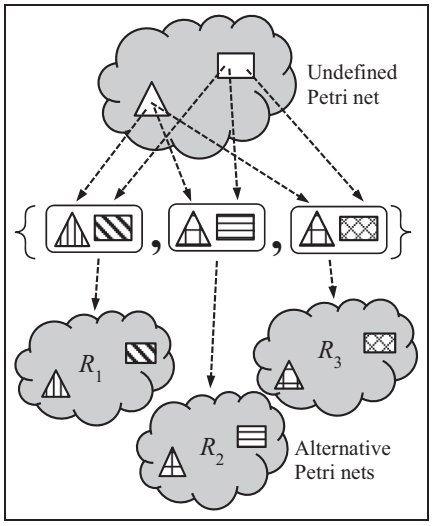

Figure 2. Alternative Petri nets.

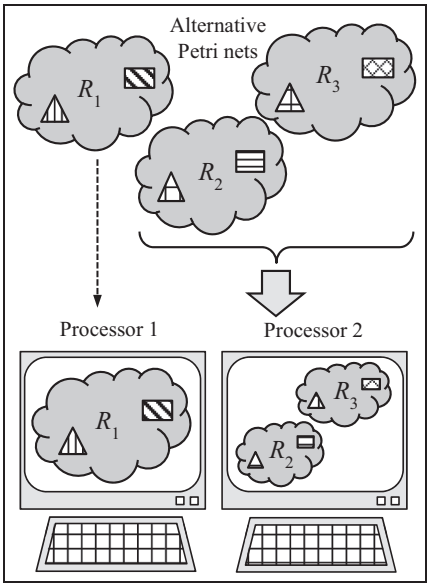

Figure 3. Distributed simulation for two available processors and sequential optimization per processor.

information are represented as geometrical shapes and the feasible values are the same shapes but filled with different patterns. The alternative configurations for the degrees of freedom can be specified by a mere combinatorial process of the feasible values associated to every degree of freedom. However, not all of the combinations are likely to be allowed; they depend on the particular undefined DES considered in the decision problem.

The choice of one structural configuration for the undefined Petri net is the purpose of the decision-making process, aimed at for the design or the efficient operation of the original DES. This choice leads to the specification of a complete model for the designed or operated system, which is called the alternative Petri net. In Figure 2, the three alternative Petri nets, $\{R_1, R_2, R_3\}$, which can be specified from the undefined Petri net illustrated in Figure 1, are depicted.

In the proposed methodology for distributed simulation presented in this paper, the decomposition of the undefined DES is not performed in submodels or parts of a given alternative structural configuration, as is the case of the approaches mentioned in the section 1.2. On the contrary, in the new methodology, the Petri net model of the DES includes the complete set of alternative structural configurations. This model is decomposed by a partition of the set of alternative structural configurations. The resulting subsets from the partition are assigned respectively to the model of the system, to fill the corresponding degrees of freedom. Every Petri net resulting from the decomposition is a complete functional model and contains one or more alternative structural configuration.

This idea can be found in Figure 3, where an intuitive representation of a decomposition of the undefined Petri net represented in Figure 1 has been performed for two available processors. The first alternative Petri net has been associated to processor 1, whereas the alternative Petri nets, 2 and 3, have been assigned to processor 2. Every processor should perform the optimization process corresponding to the assigned alternative Petri nets. In the case of processor 2, according to the paradigm of ''divide and conquer'', it is necessary to state two different optimization problems and to afford the sequential solution of both of them. The quality of the best results obtained in each solved problem will be compared with the rest of them and the best one will lead to the chosen alternative structural configuration for the undefined DES.

In the following sections, it will be shown that this sequential process for the solution of the optimization problems stated in the processors with more than a single associated alternative Petri net can be improved. In particular, it is possible to perform a single optimization for every processor, no matter the number of assigned alternative Petri nets. In order to achieve this objective, the use of an appropriate formalism, such as the alternative aggregation Petri net (AAPN), is introduced. In Figure 4, it is shown in an intuitive manner that it is possible to build a single AAPN from the two alternative Petri nets associated with processor 2.

Once a set of alternative Petri nets, associated to the same processor, has been transformed into a single AAPN, it is possible to transform the original decomposition of the undefined Petri net as it is shown in Figure 5. In this figure, a similar representation to the one depicted in the Figure 3 is presented. However, in Figure 5, processor 2 is able to perform a single optimization process instead of a sequence of processes aimed to solve the optimization problems associated to the different alternative Petri nets

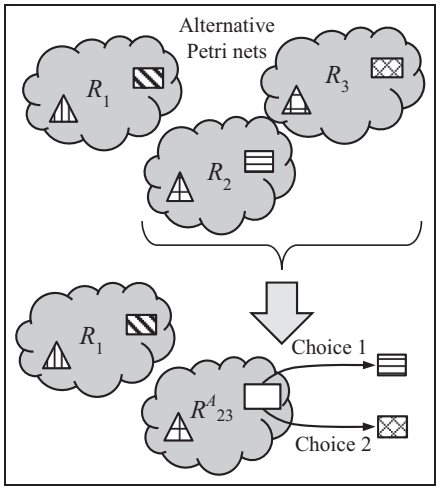

Figure 4. Transformation of the set of alternative Petri nets $\{R_1, R_2\}$ into the AAPN R^A_{23}.

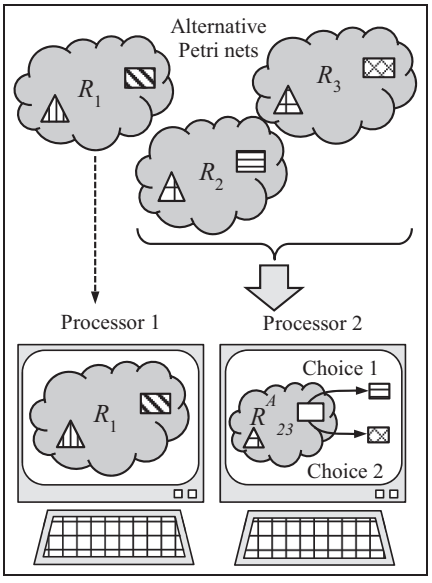

Figure 5. Distributed simulation for two available processors and single optimization per processor.

assigned to this processor. In a subsequent section, it will be shown that reducing the number of sequential optimizations may lead to a reduction in the computer requirements.

Due to the fact that the optimization performed in every processor corresponds to exclusive and independent models, the only information needed to be interchanged between them is the best solution found at the end of the process and the identification of the alternative Petri net associated with it. With this sole information, it is possible to determine the solution of the global decision problem.

The application of the mentioned methodology for the distributed simulation of the original DES can be

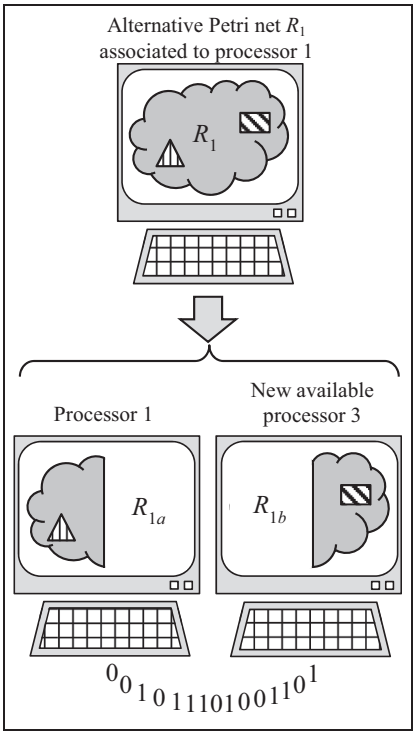

Figure 6. Complementary application of a classic approach for distributed simulation for another available processor.

complemented with other techniques, such as the ones described in section 1.2. This complementary use of different approaches for distributed simulation has been represented in Figure 6. In this figure, the alternative Petri net R_1 has been decomposed into two submodels to be simulated in two different processors: 1 and 3. In general, the decomposition of a Petri net in subnets requires time synchronization between the processors performing the simulations of the subnets. This requirement is a consequence of the concurrent evolution that is likely to exist between the subnets. The mentioned synchronization has been depicted in Figure 6 as a string of binary numbers interchanged by processors 1 and 3. The application of these standard methodologies for distributed simulation is not considered in this paper; despite this fact, a fertile collaboration among them can be envisaged.

The decision problems posed on an undefined DES with alternative structural configurations is one of the problems related to the design and operation of a DES which may be harder to solve. This fact is due to the large size of the solution space and the existence of a disjunctive constraint in the form of a set of alternative structural configurations for the model of the DES. This is the problem addressed in this paper with the purpose of reducing the computer requirements for performing the solution process.

For this objective to be achieved, the main contributions of this paper, several distributed and non-distributed

approaches are proposed, analyzed, and their relative performances are compared, making explicit the dependence of the results with the number of available processors for the distributed approaches. Furthermore, the suitability of several formalisms based on the paradigm of the Petri nets for the modeling of an undefined DES and for the implementation of a solving methodology for the associated optimization problem is tested.

The rest of the paper is organized as follows. Section 2 details the simulation-based optimization process of a DES developed for a single processor. This section consists mainly of a compilation of previous results of the authors, which have been rewritten and presented in a comprehensive manner. The rest of the sections are composed in their major parts by original contributions of the authors. Section 3 describes the theoretical background of several algorithms that allow distributing the optimization process into a number of processors. The mentioned algorithms are explained in section 4. Section 5 compares different methodologies based on the execution of simulation-based optimization algorithms in a single processor, as well as distributed algorithms on sets of processors of different sizes. Finally, section 6 presents some conclusions and open research lines.

2. Simulation-based optimization of a DES on a single processor

The design and operation of a DES require affording decision-making processes. If an efficient operation of the system is expected, the process of making decisions becomes more challenging. In this last case, which is a very usual situation for DESs, the decision-making is aimed at selecting the best possible configuration for some or all of the degrees of freedom of the system.

2.1. Paradigm of the Petri nets

In order to develop an algorithmic procedure for solving the decision-making problem, it is convenient to translate the description of the decision problem into a formal language.

In this paper, the formalism used to represent a model of the DES is based on the paradigm of the Petri nets. This section is devoted to providing an introduction to the Petri net formalism and to the dynamics of a Petri net model. Every concept presented in this section will play a role in the following sections. A deeper and broader insight into this paradigm can be found in the literature.[38–40]

Definition 1. *Marked Petri net.*

A (generalized) *marked Petri net graph* (or *Petri net system*) is a weighted bipartite graph given by the five-tuple

$$R = \langle P, T, \text{pre}, \text{post}, \mathbf{m_0} \rangle$$

where

$P = \{p_1, p_2, \ldots, p_n\}$ is the finite, non-empty, set of places (one type of node in the graph);

$T = \{t_1, t_2, \ldots, t_m\}$ is the finite, non-empty, set of transitions (the other type of node);

pre: $P \times T \rightarrow \mathbb{N}$ is the pre-incidence or input function;

post: $T \times P \rightarrow \mathbb{N}$ is the post-incidence or output function; and

$\boldsymbol{m_0}$: $P \rightarrow \mathbb{N}$ is the initial marking. □

The pre (post) function represents the directed arcs drawn from (to) any place to (from) any transition of the Petri net. The arcs are labeled by the corresponding value of the pre or post function, which is called the weight of the arc. The state of the Petri net changes with the occurrence of a transition, also called the firing, where tokens are removed from the places linked to the fired transition by incoming arcs (input places) and tokens are added to the places linked by out coming ones (output places). The number of tokens removed from or added to a given place corresponds to the weight of the arc that links the place to the fired transition. Only the enabled transitions can be fired. For a transition to be enabled, the number of tokens in the input places should be equal to or greater than the weight of the arc that links the place with the transition.

It is possible to represent the net structure by means of incidence matrices: an input or pre-incidence matrix $\boldsymbol{W}^-$ and an output or post-incidence matrix $\boldsymbol{W}^+$, both having $|P|$ rows and $|T|$ columns. As a consequence, it is possible to interpret a Petri net as a sequence of numbers or parameters that play a particular role in the model: some of them represent the static structure and some others the behavior, given by a sequence of states described by the marking of the Petri net system.[4]

The evolution of a Petri net from a state, described by its marking, by means of the firing of the transition t can be specified by the state equation

$$\mathbf{m}_k \xrightarrow{t} \mathbf{m}_{k+1} \Leftrightarrow \mathbf{m}_{k+1} = \mathbf{m}_k + \mathbf{W}(t) \cdot \mathbf{u}_t \qquad (1)$$

where $\boldsymbol{u}_t$ is the characteristic vector of t, also called the firing vector associated to t. It is defined to indicate the fact that the jth transition fires

$$\mathbf{u}_t = (0, \ldots, 0, 1, 0, \ldots 0) \qquad (2)$$

where the only 1 appears in the jth position such that $j \in \{1, \ldots, |T|\}$.

2.2. Petri net model of a DES with alternative structural configurations

Given an undefined DES and according to several considerations, such as the used formalism, the characteristics of

the DES, and the choice of the modeler, the result of the modeling process can be a single Petri net or a set of them. If the degrees of freedom are modeled as missing pieces of information in the incidence matrices of the Petri net, the alternative configurations are called structural ones and a natural and intuitive modeling process leads to a set of alternative Petri nets. Otherwise, the resulting model is normally represented by a single parametric Petri net. Please, consult Zimmerman et al.,[3] Recalde et al.,[41] Tsinarakis et al.,[37] and Latorre et al.[42] for more details.

In this paper, the objective of the decision problem will be an undefined Petri net containing several alternative structural configurations; hence, it may be represented by a set of alternative Petri nets. In this section, some definitions, related to the models of an undefined DES with alternative structural configurations will be given. A broader and slightly different presentation can be found in the work developed by Latorre-Biel et al.[43]

Definition 2. *Set of alternative Petri nets.*

Given a set of Petri nets $S_R = \{R_1, \ldots, R_n\}$, S_R is said to be a set of alternative Petri nets if $n > 1$ and $\forall i, j \in \mathbb{N}$ such that $1 \leq i, j \leq n$, and $i \neq j$. It is verified that

(i) if $\mathbf{m}(R_i) \neq \mathbf{m_0}(R_i) \Rightarrow \mathbf{m}(R_j) = \mathbf{m_0}(R_j)$;
(ii) if $\mathbf{m}(R_j) \neq \mathbf{m_0}(R_j) \Rightarrow \mathbf{m}(R_i) = \mathbf{m_0}(R_i)$; and
(iii) the incidence matrices of the Petri nets are different: $\mathbf{W}(R_i) \Rightarrow \mathbf{W}(R_j)$. □

Notice that the properties (i) and (ii) imply that R_i and R_j verify mutually exclusive evolution, that is to say, if one of them is in a state different to the initial one, the other must remain in the initial state. This property, which models the exclusive nature of the feasible choices in a decision problem, justifies the independent evolution of the different alternative Petri nets, which do not require any synchronization while the simulation is being performed in different processors.

This classic approach of having many models as feasible structural configurations can be enriched through the abstraction of some concepts, such as the exclusiveness of the evolution of the different alternative Petri nets. This abstraction leads to the definition of new formalisms, such as the AAPN.

The concept of exclusive entity is an abstraction of the models chosen to represent the different structural configurations of the DES. One particular representation of a set of exclusive entities is a set of alternative Petri nets, each one for a different structural configuration of the DES. In fact, the concept of a set of alternative Petri nets has an important role in the following definition of a set of exclusive entities.

Definition 3. *Monotypic set of exclusive entities.*

Given an DES D, a monotypic set of exclusive entities associated to D is a set $S_x = \{X_1, \ldots, X_n\}$, which verifies that

(i) the elements of S_x are exclusive, that is to say, only one of them can be chosen as a consequence of a decision;
(ii) $\forall i, j \in \mathbb{N}^*$, $i \neq j$, and $1 \leq i, j \leq n$, it is verified that $X_i \neq X_j$;
(iii) the elements of S_x are of the same type; and
(iv) $\exists$ f: $S_x \rightarrow S_R$ such that
  (a) $S_R = \{ R_1, \ldots, R_n \}$ is a set of alternative Petri nets, feasible models of D; and
  (b) f is a bijection $\Rightarrow \forall X_i \in S_x \exists !$ f$(X_i) = R_i \in S_R$ such that R_i is a feasible model for D and $\forall R_i \in S_R \exists !$ f$^{-1}(R_i) = X_i \in S_x$. □

The adjective ''monotypic'' for a set of exclusive entities refers to the fact that all of them are associated to the same type of representation, for example alternative Petri nets.

If the abstraction process continues further, an undefined Petri net with alternative structural configurations can be defined as the model of an undefined DES with alternative structural configurations.

Definition 4. *Undefined Petri net with alternative structural configurations.*

Given a DES D, an undefined Petri net with alternative structural configurations developed as a model of D, is a 10-tuple $R^U = \langle P, T, \text{pre}, \text{post}, \mathbf{m_0}, S_\alpha, S_{val\alpha}, S_x, \mathrm{R}_\gamma, \mathrm{R}_{val\alpha} \rangle$, such that $|S_x| > 1$, where

(i) S_α is the set of undefined parameters, that is to say, the variables of the Petri net that model the degrees of freedom of D;
(ii) $S_{val\alpha}$ is the feasible combination of values for the undefined parameters S_α;
(iii) S_x is a complete set of exclusive entities associated to D;
(iv) $\mathrm{R}_{val\alpha}$ is a binary relation between $S_{val\alpha}$ and S_x; and
(v) R_γ is a binary relation between S_x and S_γ, where S_γ is the set of parameters of R^U no matter if they are undefined ones or are associated to a single feasible value. □

One possible representation of an undefined Petri net with alternative structural configurations is a set of alternative Petri nets. Another particular representation of an undefined Petri net is called the AAPN.[42] As its name says, the AAPN may be constructed from the aggregation of a set of alternative Petri nets, a process described by Latorre et al.[44]. In this process, the set of exclusive entities is represented by a set of choice variables $S_A = \{a_1, a_2, \ldots, a_n\}$. See Figure 4 for an intuitive representation of this idea.

Definition 5. *Simple AAPN system.*

Given an undefined Petri net R^U, an AAPN system R^A is defined as the 10-tuple:

$$R^A = \langle P, T, \text{pre}, \text{post}, \mathbf{m}_0, S_\alpha, S_{valnstr\alpha}, S_A, \text{R}_{valnstr\alpha}, \text{f}_\text{A} \rangle$$

where

(i) S_α is a set of non-structural undefined parameters, that is to say, not contained in the incidence matrices of the net;

(ii) $S_{val\alpha}$ is the set of feasible combination of values for the undefined parameters in S_α;

(iii) S_A is the representation of the set of exclusive entities, called the set of choice variables, which verifies that $S_A \neq \varnothing$ and $S_A = \{a_1, a_2, \ldots, a_n \mid \exists! \; a_i=1, \; 1 \leq i \leq n \wedge a_j=0 \; \forall j \neq i, \; 1 \leq i \leq n \}$, where the choice of $a_i=1$ is the result of a decision;

(iv) $\text{R}_{val\alpha}$ is a binary relation between $S_{val\alpha}$ and S_A; and

(v) $f_A: T \rightarrow f(a_1, \ldots, a_n)$ assigns a function of the choice variables in regard to each transition t such that type$[f_A(t)]$ = Boolean. □

Notice that the adjective ''simple'' applied to the AAPN in the previous definition is a reference to the fact that R^A does not have any undefined structural parameter. For a more detailed description of the AAPNs and the transformation algorithm from a set of alternative Petri nets to an equivalent AAPN, see Latorre et al.[44]

2.3. Statement of an optimization problem

Once the model of the original DES has been developed, it is necessary to complete the formal statement of the decision problem with additional constraints in the form of inequalities (for example a minimal production that should be complied to in the case of a manufacturing facility or a maximal makespan).

In general, a decision problem requires the choice of the best solution according to a certain criterion. In this case, it is necessary to quantify this criterion, usually in the form of one or several performance measurements, which are included in an objective or multi-objective function $f(x)$. This function should also be included in the formal statement of the optimization problem associated to the decision problem.

In sections 2.5 and 2.6, several formal statements of optimization problems of interest in this paper will be presented.

2.4. Influence of the choice of the modeling formalism on the computer requirements for solving an optimization problem

If the formalism chosen to model an undefined DES is a set of n alternative Petri nets, $S_R = \{R_1, R_2, \ldots, R_n\}$, the subsequent optimization problem can be solved by means of a classic approach. This methodology is based in the concept of ''divide and conquer'', where the optimization process is divided into a set of n simpler optimization problems. Each one of the simple problems is associated to a single one of the alternative Petri nets of S_R. According to this idea, the search for the best solution to the problem is decomposed into n independent search processes, focused on smaller solution spaces than the one that corresponds to the complete set S_R. In particular, the ith given independent search is devoted to the solution space of the associated alternative Petri net R_i.

If this classic methodology is executed in a single processor, it is necessary to solve sequentially a set of n optimization problems based on a single alternative Petri net.

It is possible to improve the efficiency of the methodology to solve the original decision problem by using a more compact and appropriate model for the alternative structural configurations of the DES or, what is the same, for the associated set of exclusive entities. This improvement is based on several facts pointed out in the following paragraphs:

(a) In real systems, it is very common that the different alternative Petri nets share some subnets, which correspond to common physical subsystems. In particular, there are cases where the diverse alternative structural configurations are built from the different combinations of the same subsystems. Thanks to this idea, the model of the system might be greatly reduced by the removal of this redundant information.

(b) The state-space of a set of alternative Petri nets is isomorphous to the union of the state-spaces of the alternative Petri nets. This property is not verified by a set of Petri nets that are subsystems of the same model instead of being alternative structures for it, since the former leads to a state-space for the set of Petri nets that is in general much larger than the union of the state-spaces of the original Petri nets. In fact, the set of Petri nets may present states that are feasible combinations of the states of the different Petri nets, while the alternative Petri nets verify the mutually exclusive evolution.

(c) If the optimization is reduced to a single process, there exists only one solution space to be explored, whereas there are n different solution spaces associated to the n simpler optimization problems arising from the approach of ''divide and conquer''. When the search of the most promising solutions to be evaluated is performed by means of a heuristic, the computer resources are focused on the most promising regions of the solution space, while the ''divide and conquer'' approach scatters the computer resources into the n search processes of the different solution spaces.

The improved methodology to solve the optimization problem, for the moment in a single processor, is based in the use of a single and compact model, based on the formalism of the AAPN. This model may require less data to be described than an original set of alternative Petri nets if the degree of similarity between the nets is high. This degree of similarity is related to a large number of coincident elements in the incidence matrices of the alternative Petri nets and may correspond to alternative structural configurations with shared subnets.

The performance of the simulation of the evolution of a single alternative Petri net is likely to be faster than a simulation of the AAPN. Nevertheless, better use of the computer resources in the exploration of the complete solution space may lead to an optimization process much shorter in the case of the AAPN than the set of alternative Petri nets (see paragraph (c) in this section).

Summarizing, two optimization methodologies are introduced in this section. They are applied to a general decision problem based on an undefined DES and they are based on different formalisms for the development of the model of the DES. As a result of the considerations of this section, it is expected that on a single processor, the approach based on an AAPN outperforms the methodology inspired in the paradigm of ''divide and conquer''. In section 2.5, a deeper insight into both methodologies will be provided.

2.5. Optimization strategy based on the paradigm of "divide and conquer"

The first optimization methodology to be considered in detail is a classic approach based on the concept of ''divide and conquer''. The corresponding optimization problem contains a set of alternative Petri nets as a model of the undefined Petri net and can be stated as follows:

Definition 6. *Optimization problem based on a set of n alternative Petri nets.*

Optimize $f(x)$, subject to

(i) $S_R = \{R_1, R_2, \ldots, R_n\}$ is a model of the system in the form of a set of alternative Petri nets; this is a disjunctive constraint;
(ii) additional constraints that can be written in the form
  (a) $g_j(x) > 0, 1 \leq j \leq n_{gj}$,
  (b) $h_k(x) = 0, 1 \leq k \leq n_{hk}$; and
(iii) $x \in D_{om}$, solution space, where (x, R_i) is the solution of the optimization problem and contains all the undefined parameters of the model: $x = (\alpha_1, \alpha_2, \ldots, \alpha_p)$. □

This problem can be decomposed into n independent optimization subproblems of the form:

Definition 7. *Optimization subproblem based on an alternative Petri net.*

Optimize $f_i(x_i)$, subject to

(i) $R_i \in S_R = \{R_1, R_2, \ldots, R_n\}$ is a partial model of the system given by a single feasible combination of values for the undefined structural parameters in the form of an alternative Petri net; it is not a disjunctive constraint;
(ii) additional constraints, written in the form
  (a) $g_{ij}(x_i) > 0, 1 \leq j \leq n_{gij}$,
  (b) $h_{ik}(x_i) = 0, 1 \leq k \leq n_{hik}$; and
(i) $x_i \in D_{omi}$, solution space, where x_i contains all the undefined parameters of R_i and is the solution of the ith optimization subproblem. □

In the standard approach of ''divide and conquer'', the optimization strategy consists of solving independently the n different optimization subproblems associated to every alternative Petri net, obtaining the best $f_i(x_i)$ in each case, comparing them, and choosing the best $f_{opt}(x_i)$ of all. $f_{opt}(x_i)$ corresponds to a configuration of the undefined parameters of the Petri net model given by x_i and, hence, to the solution of the associated decision problem.

2.6. Optimization strategy based on an AAPN

The second optimization strategy to be considered in detail is based in the formalism of the AAPN (see definition 5) to represent the same undefined DES considered in the previous section. Under this approach it is possible to translate the original decision problem based on a DES into the following optimization one:

Definition 8. *Optimization problem based on an AAPN.*

Optimize $f(x)$, subject to

(i) R^A is a model of the system in the form of an AAPN; this is a disjunctive constraint;
(ii) additional constraints that can be written in the form
  (a) $g_j(x) > 0, 1 \leq j \leq n_{gj}$,
  (b) $h_k(x) = 0, 1 \leq k \leq n_{hk}$; and
(iii) $x \in D_{om}$, solution space, where (x, a) is the solution of the optimization problem that contains all the undefined parameters of the model $x = (\alpha_1, \alpha_2, \ldots, \alpha_p)$. □

The use of an AAPN model allows performing a single optimization instead of the n processes associated to the classic approach. Moreover, the size of the model may contain less data in the AAPN representation than in the whole set of alternative Petri nets as a consequence of the removal of redundant information mentioned in paragraph (a) of section 2.4.

On the other hand, the incidence matrices W^+ and W^- associated to the AAPN may be larger than the incidence matrices of any of the alternative Petri nets;[42] hence, the application of equation (1) to calculate a step in the evolution of a Petri net requires more computational resources in the case of the AAPN than a single alternative Petri net. Nevertheless, the optimization problem based on a set of alternative Petri nets requires n optimization processes to be solved.

Moreover, each optimization process, associated to an alternative Petri net, explores a region of the total solution space, which implies a waste of time in the less promising regions; whereas, the AAPN approach explores the solution space as a whole avoiding the less promising areas and devotes the computational resources to where the heuristic guides the search to the regions that are the most promising ones.

For a more detailed description on a particularly successful heuristic to solve the mentioned optimization problems, the genetic algorithms, see Latorre et al.[4]

2.7. *Example of the design of a manufacturing facility*

Latorre et al.[44] presents a case-study, hereafter referred to as example 1, where a manufacturing facility in the process of being designed is considered for decision-making. In this application case, there are two alternative structural configurations that include respectively a flexible subsystem and a non-flexible one for the quality control, assembly of parts, and packing of the final products. In each one of the alternative configurations, there are other degrees of freedom, such as the presence and size of intermediate buffers, the layout of several machining centers, and the number of robots to perform certain tasks in the facility.

Figure 7 shows an intuitive representation of the decision problem. The original DES is a manufacturing facility in the process of being designed. After a modeling process, an undefined Petri net is obtained. The degrees of freedom of the model have been represented in the following way. The undefined structural parameters, which correspond to the two different quality control, packing, and assembly systems, are represented by a triangle. It has two feasible configurations: *a* and *b*. On the other hand, the undefined non-structural parameters, such as the size of intermediate buffers or the number of robots are represented by a rectangle. The feasible combination of values for the undefined non-structural parameters are represented by the natural numbers 1, 2, 3, …

The undefined DES is modeled in two ways: as a set of two alternative Petri nets and as an AAPN. It has to be said that with the exception of the mentioned subsystem of quality control, assembly, and packing, the structure of the rest of the facility is the same in both cases. The existence of shared subnets between the structural alternatives is very common in the design of a DES.[41]

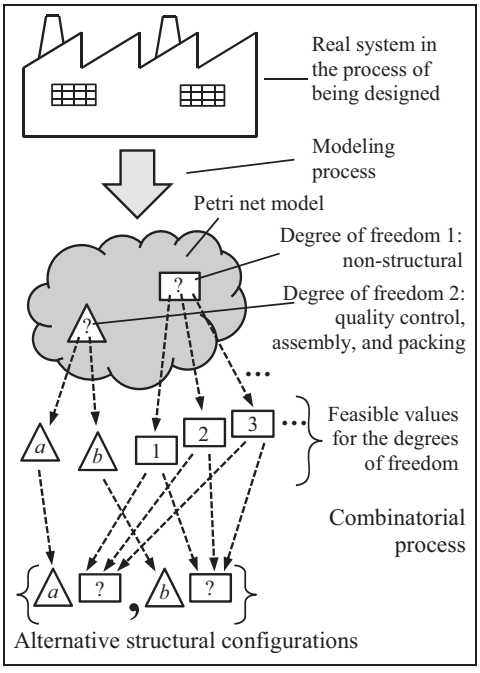

Figure 7. Undefined Petri net of example 1.

In the mentioned paper, a comparison is described between two optimization approaches. On the one hand, a ''divide and conquer'' approach in the form of two optimization problems, each one of them associated to one of the alternative Petri nets, is presented. On the other hand, the search in a single solution space associated to an AAPN model is described. The comparison takes into account two performance parameters from both approaches: the quality of the best solution obtained, measured by the value of a multi-objective function, as well as the computer requirements, measured by the time required to finish the complete optimization process.

The results of the comparison show that the quality of the solutions is comparable and the time required to finish the process based on the AAPN is half the time needed by the classic approach. With this case-study it has been shown that at least in this example the approach based on an AAPN is very efficient when compared to the classic ''divide and conquer''. □

Other application cases, studied by the authors, with a larger number of alternative Petri nets lead to a trend showing that the larger the number of shared subnets, the more efficient the AAPN is compared to the set of alternative Petri nets. The previous considerations, including the already mentioned improvements of the optimization based on an AAPN and their justifications, may lead to the conclusion that the AAPN is a formalism that can improve the efficiency of the decision-making processes for an undefined DES, performed on a single processor.

3. Distributed computation to solve a simulation-based optimization

In a distributed computing system (DCS), programs and data files are distributed among several processing units which may contribute to the simultaneous execution of a single program or, alternatively, different programs can be executed concurrently.

Several authors have addressed the simulation of the evolution of a Petri net using distributed algorithms. These algorithms aim to reduce the execution time by distributing the computational effort into a number of processors. As it has already been mentioned in section 1, most of these strategies are compatible with the approach that is presented in this paper; thus, interesting research work might be afforded by the combination of some of these techniques.

3.1. Definition of optimization subproblems of size m

In this section, a methodology to distribute the solving procedure of the optimization problems described in definition 6 and definition 8 into a number of different processing units will be described. This methodology is applied by decomposing the optimization problem into autonomous subproblems that minimize the computational effort, as well as the information interchanged between the processors. Every subproblem is assigned to a different processor to be solved.

The definition of a subproblem of size m can be performed in the following way:

Definition 9. *Optimization subproblem of size m based on a subset of alternative Petri nets.*

Given an optimization problem based on $S_R = \{R_1, R_2, \ldots, R_n\}$ (see definition 6), an optimization subproblem of size $1 \leq m \leq n$ based on $S_{Ri} \subseteq S_R$ can be defined in the following way

Optimize $f_i(x_i)$, subject to

(i) $S_{Ri} \subseteq S_R = \{R_1, R_2, \ldots, R_n\}$, $|S_{Ri}| = m \leq n$;
(ii) additional constraints that can be written in the form
  (a) $g_{ij}(x_i) > 0, 1 \leq j \leq n_{gij}$,
  (b) $h_{ik}(x_i) = 0, 1 \leq k \leq n_{hik}$; and
(iii) $x_i \in D_{omi}$, solution space, where x_i is the solution of the optimization problem that contains all the undefined parameters of S_{Ri}. □

In Figure 3, the Petri net models of different statements of optimization problems have been represented. In the upper part of Figure 3, a set of three alternative Petri nets is shown. This set of alternative Petri nets is associated to an ''optimization problem based on a set of n=3 alternative Petri nets'', as described in definition 6. In the bottom-left corner of Figure 3, a single alternative Petri net, belonging to an ''optimization subproblem based on an alternative Petri net'' is shown, as described in definition 7. This subproblem has been assigned to processor number 1. Finally, in the bottom-right corner of the Figure 3, a set of two alternative Petri nets, associated to an ''optimization subproblem of size m=2 based on a subset of alternative Petri nets'', can be found. This subproblem has been assigned to processor number 2.

In an analogous way, it is possible to state another definition of the same optimization problem, based on a different representation of the model of the undefined DES. In this case the chosen formalism is the AAPN:

Definition 10. *Optimization subproblem of size m based on an AAPN.*

Given an optimization problem based on $S_R = \{R_1, R_2, \ldots, R_n\}$ (see definition 6), an optimization subproblem of size $1 \leq m \leq n$ based on an AAPN R_i^A, equivalent to $S_{Ri} \subseteq S_R$, where $|S_{Ri}| = m \leq n$. ,can be defined in the following way

Optimize $f_i(x_i)$, subject to

(i) R_i^A, AAPN equivalent to S_{Ri};
(ii) additional constraints that can be written in the form
  (a) $g_{ij}(x_i) > 0, 1 \leq j \leq n_{gij}$,
  (b) $h_{ik}(x_i) = 0, 1 \leq k \leq n_{hik}$; and
(iii) $x_i \in D_{omi}$, solution space, where x_i is the solution of the optimization problem that contains all the undefined parameters of R_i^A. □

In Figure 5, an intuitive representation of an AAPN, associated to an ''optimization subproblem of size m=2 based on an AAPN'' has been shown. In the bottom-right corner of Figure 5, an intuitive representation of this AAPN, assigned to processor number 2, can be seen.

By means of the application of a transformation algorithm from a set of alternative Petri nets, S_{Ri}, into an equivalent AAPN, R_i^A, detailed by Latorre et al.[44] Given a subproblem described by definition 9, it is always possible to obtain an equivalent subproblem described by definition 10.

3.2. Decomposition of an optimization problem into subproblems

Given an instance of a problem described by definition 6, ''optimization problem based on a set of n alternative Petri nets'', it is always possible to decompose it into k non-redundant subproblems, where $k \leq n$. For this task to be performed, a decomposition of the set of alternative Petri nets $S_R = \{R_1, R_2, \ldots, R_n\}$ is afforded, such that the following properties are verified:

(i) $S_R = S_{R1} \cup S_{R2} \cup \ldots \cup S_{Rk}$: this property is justified by the fact that every feasible alternative configuration of the DES should be included in one of the subproblems resulting from the decomposition;
(ii) $S_{Ri} \neq \varnothing$, $1 \leq i \leq k$: it does not make sense to define a subproblem, which does not include any alternative configuration of the undefined DES to be analyzed; and
(iii) $S_{Ri} \cap S_{Rj} = \varnothing$, $1 \leq i,j \leq k$: this last property is related to the need to prevent a repetitive analysis of a given alternative configuration by ensuring that any of the configurations is associated to a single subproblem.

The previous three properties, verified by the decomposition of $S_R = \{R_1, R_2, \ldots, R_n\}$ into subproblems, define a partition of S_R.

It is clear that, in general, different decompositions of an ''optimization problem based on a set of n alternative Petri nets'' (see definition 6) would perform differently when the same solving technique is applied. For this reason, it is convenient to know all the range of possibilities that this decomposition approach offers.

The power of this approach can be measured partially in terms of the number of possible decompositions. As it will be shown in the following paragraphs, in a general case, it is possible to find a large number of feasible decompositions into subproblems of a given optimization problem. This result allows searching for the partition of S_R that leads to the most efficient optimization process by distributed computation:

Proposition 1. *Number of possible decompositions of an optimization problem based on a set of n alternative Petri nets, S_R.*

Given an instance of the optimization problem described in definition 6, where $|S_R| = n$, the number of feasible decompositions into non-redundant subproblems is given by the Bell number B_n.

Proof

The decomposition of the problem consists of dividing it into subproblems that include the elements corresponding to the assigned feasible configurations: objective function, alternative Petri net or nets, additional constraints, and domain.

To avoid loss of information, to permit consistency in the subproblems, and to prevent redundancy the following three properties should be complied with

(i) $S_R = S_{R1} \cup S_{R2} \cup \ldots \cup S_{Rk}$;
(ii) $S_{Ri} \neq \varnothing$, $1 \leq i \leq k$; and
(iii) $S_{Ri} \cap S_{Rj} = \varnothing$, $1 \leq i,j \leq k$.

These three properties specify the decomposition of S_R as a partition.

According to Rota,[45] the number of feasible partitions of a set of cardinality n can be calculated by the Bell number B_n. □

The sets S_{Ri}, $1 \leq i \leq k$, obtained from the partition of S_R are called blocks, classes, or cells by different authors.

The Bell number B_n can be obtained by the following recursion formula[45]

$$B_n = \sum_{i=0}^{n-1} \binom{n-1}{i} \cdot B_i \tag{3}$$

where B_0=1 by convention[45] and the binomial coefficients can be calculated by different means, for instance

$$\binom{n-1}{i} = \frac{(n-1)!}{i! \cdot (n-i-1)!} \text{ for } 0 \leq i \leq n-1 \tag{4}$$

The Bell numbers are also called exponential numbers[46] and grow very rapidly with the cardinality of the set S_R, which implies the advantage of having in general a large number of possible decompositions for an optimization problem based on a set of alternative Petri nets.

To provide an idea of the rate at which their value is increased, some Bell numbers have been listed in Table 1 as an example.

The interpretation of these Bell numbers in the context of this paper can be seen by the following example. Let us consider an undefined Petri net with 100 alternative structural configurations, described by S_R, such that $|S_R| = 100$. The number of feasible decompositions of an optimization problem based on S_R into a set of k processors, where $1 \leq k \leq n$ =100, is $4.75853912767648 \cdot 10^{115}$. Among them, there will be some decompositions that may lead to very efficient solution processes of the optimization problem.

Table 1. Value of some selected Bell numbers.

$B_0 = 1$
$B_1 = 1$
$B_2 = 2$
$B_3 = 5$
$B_4 = 15$
$B_5 = 52$
$B_6 = 203$
$B_7 = 877$
$B_8 = 4140$
$B_9 = 21147$
$B_{10} = 115975$
$B_{15} = 1382958545$
$B_{25} = 4638590332230000000$
$B_{50} = 1.85724268771078 \cdot 10^{47}$
$B_{100} = 4.75853912767648 \cdot 10^{115}$

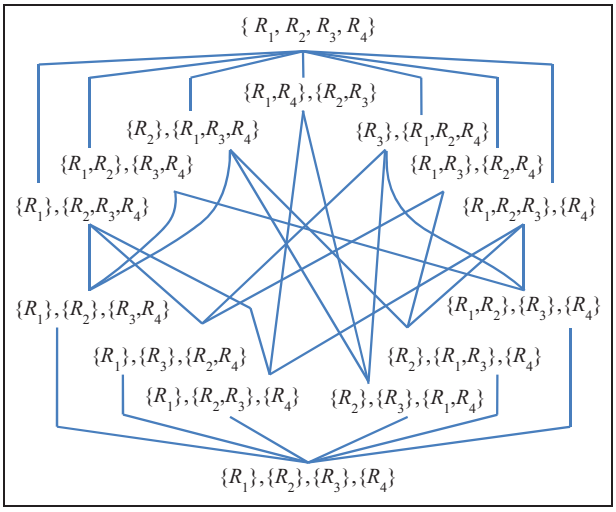

Figure 8. Hasse diagram representing the feasible partitions of a set of four alternative Petri nets.

3.3. Example of decomposition of an optimization problem into subproblems

As example 2, let us consider the set of $n = 4$ alternative Petri nets $S_R = \{R_1, R_2, R_3, R_4\}$ included in the statement of an optimization problem according to definition 6.

This problem can be decomposed in $B_4 = 15$ different ways, represented by the Hasse diagram depicted in Figure 8.

In the previous Hasse diagram, the different $B_4 = 15$ possible partitions of S_R, structured in four levels according to the order of the partition, can be seen. This order describes the number of resulting blocks of alternative Petri nets from the partition which in this case ranges from 1 to 4. Notice that for a given partition, a block is represented by a set of alternative Petri nets between curly brackets. In the distributed scheme presented in this paper, every block corresponds to the set of alternative Petri nets associated to an optimization subproblem assigned to a given processor. As a consequence, the number of available processors will correspond to the order of the partition considered for the case.

Furthermore, the lines in the diagram link sequences of decompositions, which increase the order of the resulting set of blocks from the top of the diagram to the bottom or decrease it in the opposite direction.

Let us consider now a case, where there are $k = 3$ available processors for executing the optimization algorithm mentioned in the last example (see definition 6). In this situation, the partition of S_R should be made of order $k = 3$ in order to distribute the alternative Petri nets into three subproblems, for solving each one of them in a different processor. As it can be seen in the Hasse diagram, the number of feasible partitions of order 3 is 6. Hence, it is necessary to use an additional criterion to choose one of the six available partitions. Eventually, the selected partition will describe how to distribute the alternative Petri nets into subproblems that will be assigned to different processors. More general considerations will be developed and detailed in the following paragraphs.

The type of a partition can be considered as notation for designing the partitions of a set. The arrangement of the sizes of the blocks into a decreasing sequence is called the type of a partition. For example, the partition of order 1 is of type 1. The partitions of order 2 are of type 22 ($\{R_a, R_b\}$, $\{R_c, R_d\}$) and of type 31 ($\{R_a\}$, $\{R_b, R_c, R_d\}$). The partitions of order 3 are all of type 211, while the partition of order 4 is 1111. □

3.4. Decompositions of an optimization problem for k available processors

Let us consider now a set of k equivalent processor systems, which are available for a distributed optimization. Notice that the use of processors of different characteristics, for performing a balanced distribution of tasks, has not been considered in this paper. It is our interest to decompose the optimization problem based on a set of n alternative Petri nets S_R, into k tasks to be executed in the processors.

The following different cases may arise according to the values of n and k:

(a) $k = n$: in this case, n subproblems can be stated according to definition 7. Each subproblem is associated with a different processor;
(b) $k > n$: consider case (a) and divide some of the subproblems of size 1 into subprocesses, by means of other techniques of distributed simulation mentioned in section 1.2;
(c) $k = 1$: a single problem, according to definition 6 or definition 8, can be posed; and
(d) $1 < k < n$: this situation is described below, which consists of distributing an optimization problem based on n alternative Petri nets into a set of k processor systems.

In real applications, the case described in (d) is very common, since the number k of available processors, a scarce resource in general, is usually smaller than the number of alternative structural configurations for an undefined DES. For this reason, the following paragraphs will be devoted to this case.

3.5. Calculation of the number of feasible decompositions of an optimization problem for a given k and n

This section will address a first step in the development of an algorithm to solve an optimization problem, as

Table 2. Values of the Stirling numbers of the second kind from n=1 to n=10.

n\k	1	2	3	4	5	6	7	8	9	10	B_n
1	1										1
2	1	1									2
3	1	3	1								5
4	1	7	6	1							15
5	1	15	25	10	1						52
6	1	31	90	65	15	1					203
7	1	63	301	350	140	21	1				877
8	1	127	966	1701	1050	266	28	1			4140
9	1	255	3025	7770	6951	2646	462	36	1		21147
10	1	511	9330	34105	42525	22827	5880	750	45	1	115975

described in definition 6, for a given pair (n, k). Notice that n is the size of the set of alternative Petri nets, whereas k is the number of available processors. This first step consists of calculating the number of feasible decompositions of the problem. Once this figure is known, the choice of the best partition of the set of alternative Petri nets S_R will be afforded.

Proposition 2. *Number of possible decompositions of an optimization problem based on S_R into a set of k processor systems.*

Given an optimization problem, as described in definition 6, where $|S_R| = n$, the number of feasible decompositions into k non-redundant subproblems is given by the Stirling number of the second kind $S(n, k)$.

Proof

The decomposition of the original optimization problem divides it into k subproblems. In particular, the set of n alternative Petri nets that model the undefined DES is distributed into the k available processors. In addition to the subset of alternative Petri nets, each one of the k subproblems includes additional elements, such as an objective function, a set of additional constraints, and the domain of the components of the feasible solutions.

To avoid loss of information, to permit consistency in the subproblems, and to prevent redundancy in the division of S_R, this decomposition should be a partition, as stated in the proof of proposition 1.

The number of feasible partitions of a set of cardinality n into k non-empty blocks can be calculated by the Stirling number of the second kind $S(n, k)$.[47] □

The Stirling number $S(n, k)$ can be calculated by the following recurrence relation[47]

$$S(n,k) = k \cdot S(n-1,k) + S(n-1,k-1) \qquad (5)$$

where $n, k \geq 1$, and by convention $S(0, 0)=1$, $S(n, 0)=0$ for $n \geq 1$, $S(n, k) = 0$ for $k > n$.

Alternatively, it is possible to represent explicitly $S(n, k)$ as a finite sum[47] and respectively

$$\begin{aligned} S(n,k) &= \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n \\ &= \frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n \end{aligned} \qquad (6)$$

where $0 \leq k \leq n$.

The lowest Stirling numbers of the second kind are represented in Table 2. In the position that corresponds to the ith row (number n) and the jth column (number k) the Stirling number $S(i, j)$ has been represented.

In example 2, it is possible to calculate the Stirling numbers of the second kind $S(n, k)$ such that $1 \leq k \leq n = 4$. The order of the partition corresponds to the second argument of the Stirling number of the second kind k. As a consequence it is possible to calculate in example 2 the following numbers:

* number of partitions of order 1: $S(4, 1) = 1$;
* number of partitions of order 2: $S(4, 2) = 7$;
* number of partitions of order 3: $S(4, 3) = 6$; and
* number of partitions of order 4: $S(4, 4) = 1$.

In the Table 2, it can also be seen, for some cases, the relation existing between the Bell numbers and the Stirling numbers of the second kind, which can be explicitly represented as

$$B_n = \sum_{k=0}^{n} S(n,k)\mathrm{m} \qquad (7)$$

This relation is immediately obtained, since the number of ways an n-element set can be partitioned is the sum of the number of ways in which this set can be partitioned into exactly k blocks ranging from 0 to n.

3.6. *Choice of the decomposition of an optimization problem for a given* k *and* n

Once the number of feasible decompositions of an optimization problem into k non-redundant subproblems has

been calculated, it is convenient to determine which one of the feasible partitions should be chosen to afford the distribution of the subproblems into the available k processors.

The criterion to choose one of the feasible partitions of S_R can be different regarding the final statement of the optimization problem. It is possible to state a collection of k subproblems of the types mentioned in the following:

(a) subproblems associated to a subset of alternative Petri nets, according to definition 9 and definition 7;
(b) subproblems associated to an AAPN or a single alternative Petri net, according to definition 10 and definition 7, respectively; and
(c) a combination of subproblems associated to a subset of alternative Petri nets or an AAPN, according to definitions 7, 9, and 10.

The following considerations address some criteria to choose successfully a certain partition for the alternative structural configurations of an undefined DES in the three cases described in the previous paragraphs. One successful criterion in the case described in paragraph (a), for choosing one of the feasible partitions of S_R, may be to construct blocks of similar size in order to balance the computational requirements of the subproblems. It is convenient to group together the fastest subproblems in the largest blocks in order to speed up the complete distributed optimization. In general, due to the fact that the most demanding operation in the optimization process is the simulation of the evolution of the Petri net by the application of equation (1), the fastest subproblems are characterized by the smallest incidence matrices. An appropriate criterion associated to the case described in paragraph (b) may be to obtain blocks of similar size and group together with the alternative Petri nets that share subnets. At a subsequent step, every block of alternative Petri nets can be transformed into an AAPN according to the algorithm described in Latorre et al.[44]. The Petri nets that share subnets will lead to an AAPN, where the shared subnet appears only once; hence, this redundant information is removed from the model of the system. The result is a model described by a smaller amount of data, which would be processed by the optimization algorithm in an efficient way. A successful criterion associated to the last option, described in paragraph (c), may be a combination of the previously mentioned ones.

In this section, the definitions of some optimization subproblems have been stated. Furthermore, two propositions have been proven regarding the calculation of the feasible decompositions of a given problem into feasible subproblems. Finally, some criteria to choose one of the feasible decompositions to define the solving strategy have been presented.

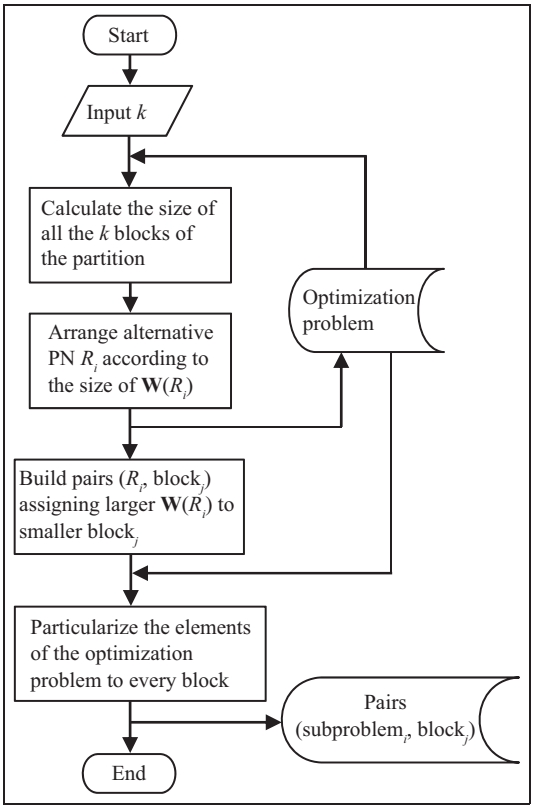

Figure 9. Algorithm 1.

In section 4, specific algorithms to distribute the tasks to different processors, to describe the optimization tasks themselves, and to transform an optimization subproblem based on a set of alternative Petri nets into a more efficient one, based on an AAPN, are discussed.

4. Algorithm for a distributed solution of an optimization problem

In this section, some algorithms for the distribution of the problem into a number of processors, for the conversion of a subset of alternative Petri nets into an AAPN and for the execution of the optimization process are presented.

Algorithm 1, represented in Figure 9, describes the process of distributing an optimization problem based on a set of n alternative Petri nets $S_R = \{R_1, R_2, \ldots, R_n\}$ into a set of k processor systems. The resulting subproblems are aimed to be associated to sets of alternative Petri nets, as stated in definition 9 and definition 7.

Algorithm 1 requires as input data the number of available processors k, as well as the definition of the optimization problem, including the objective function, the model of the undefined DES in the form of a set of alternative Petri nets S_R, the additional constraints, and the domains of the components of the solution of the problem.

Algorithm 1

Step 1. Calculation of the size of k blocks of the partition of S_R in terms of the number of alternative Petri nets. In order to balance the size of the blocks, the distribution of alternative Petri nets should be done in a way that makes the size of every block as small as possible.
Step 2. Make an ordered list of alternative Petri nets R_i according to the size of their incidence matrix $\mathbf{W}(R_i)$.
Step 3. Assign the alternative Petri nets of larger incidence matrix to the smaller blocks.
Step 4. Particularize the other components of the optimization problem to the blocks created in the partition of S_R. Every subproblem is stated and assigned to a different block. Every block is assigned to a different available processor. □

This algorithm tries to compensate for the computational requirements of every subproblem by gathering the most demanding alternative Petri nets, which are the ones with larger incidence matrices, into the smallest subsets of the alternative Petri nets, which are the ones associated to the smallest blocks.

Algorithm 2, depicted in Figure 10, is focused on the process of distributing an optimization problem based on a set of n alternative Petri nets into a set of k processor systems. The resulting subproblems are aimed to be associated to an AAPN or a single alternative Petri net if the size of the corresponding block is 1, as stated in definition 10 and definition 7.

Algorithm 2

Step 1. Calculate the size of k blocks of the partition of S_R, making them as small as possible. Notice that k is the number of available processors.
Step 2. Make an ordered list of alternative Petri nets according to the size of their incidence matrix.
Step 3. Assign the alternative Petri net in the list with the larger incidence matrix to the smaller empty block of the partition. Remove the Petri net from the ordered list.
Step 4. While there is room free in the block: Search for the Petri net from the list with the largest subnet shared with the previous one. Add it to the block. Remove it from the list. Repeat Step 4 until there is no room free in the block.
Step 5. If the ordered list of alternative Petri nets list is not empty, go to step 3.
Step 6. Particularize the other components of the optimization problem to the blocks created in the partition of S_R. Every subproblem is stated and assigned to a different block. Every block is assigned to a different available processor. □

The previous algorithm groups in the same blocks the alternative Petri nets that share the largest blocks. Algorithm 2 reaches a compromise between speed of computation and result. In fact, it would be better to find subnets shared by more than a single Petri net. Another interesting option consists of developing a dynamic grouping of alternative Petri nets into blocks, whose size would be increased as new, large subnets are found to be shared by the Petri net that already belongs to the block. Nevertheless, the very demanding computer requirements needed for performing this task has led to this

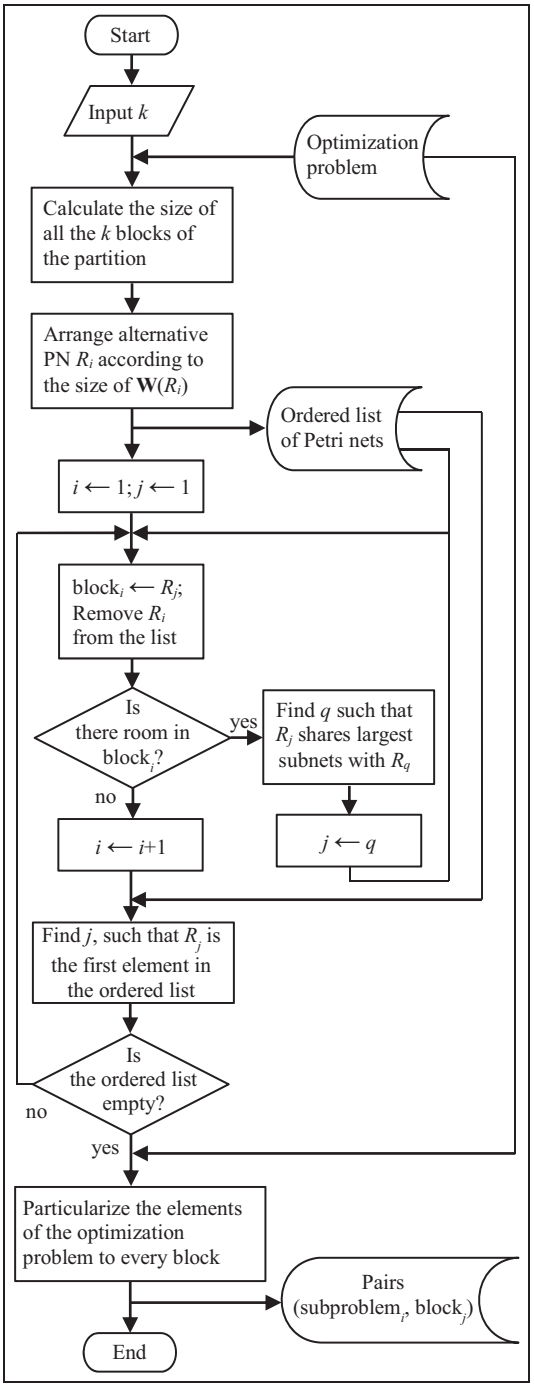

Figure 10. Algorithm 2.

more modest algorithm 2. This algorithm focuses on assigning a Petri net to each processor and to grouping it with other alternative Petri nets remaining to be assigned and that show similarity with respect to the previous one. This measurement of similarity is expected to lead in many real cases to a significant reduction in the redundant information contained in the original set of alternative Petri nets, developed as a model of the undefined DES.

Another interesting approach to afford this distribution of alternative Petri nets into subproblems can be done by identifying the common real subsystems, which usually lead to shared subnets, as a criterion to group the alternative Petri net models. Experience shows that this approach is very efficient.[42,44]

Algorithm 3 addresses the following step after the execution of algorithm 2. Once the alternative Petri nets of the original statement of the optimization problem are distributed into k blocks to be implemented in k processors, algorithm 3 transforms the subsets of the alternative Petri nets, belonging to every block, into AAPNs. See Latorre et al.[42] for more details.

Algorithm 3, including a subroutine which describes the transformation of the set of alternative Petri nets of a given subproblem into a single AAPN, is presented in Figure 11. This subroutine is presented in Figure 12.

Notice that algorithm 3 may be easily transformed to be executed in a distributed way by solving the transformation of block$_j$ in processor j. The number of blocks of the partition of S_R has been made to coincide with the number of available processors k.

Algorithm 3

Step 1. Determine the first block as the current block to transform.

Step 2. Check the size of the current block. If the size is 1, that is to say, it contains a single alternative Petri net, then determine the following block as the current block and repeat step 2 again.

Step 3. Call the subroutine from algorithm 1 in order to transform the set of alternative Petri nets associated with the current block into a single AAPN.

Step 4. Determine the following block as the current block, unless the final block$_k$ has already been transformed, and go to step 2. □

Subroutine for algorithm 3

Step 1. Divide the alternative Petri nets of block$_j$ into subnets, trying to find shared subnets.

Step 2. Take the first alternative Petri net as the seed of the AAPN of the jth block.

Step 3. For every ith alternative Petri net, add to the seed of the AAPN the subnets and link transitions of the ith

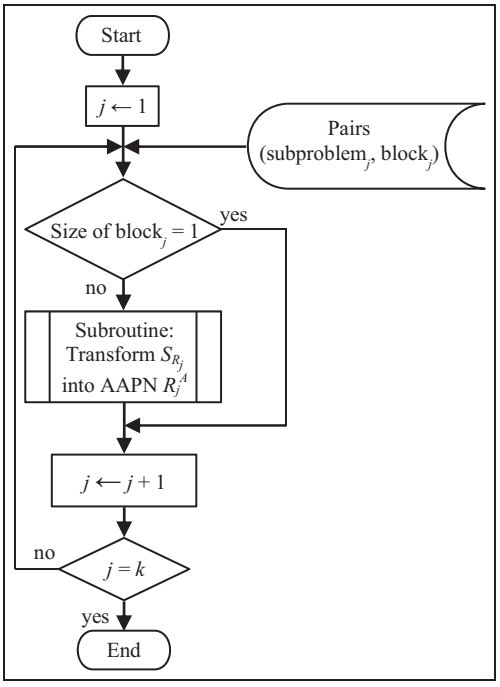

Figure 11. Algorithm 3.

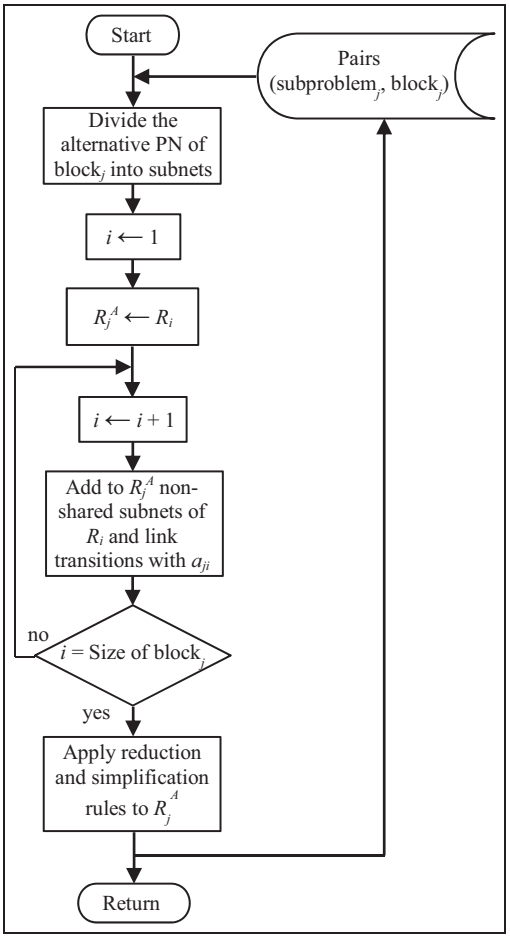

Figure 12. Subroutine for Algorithm 3.

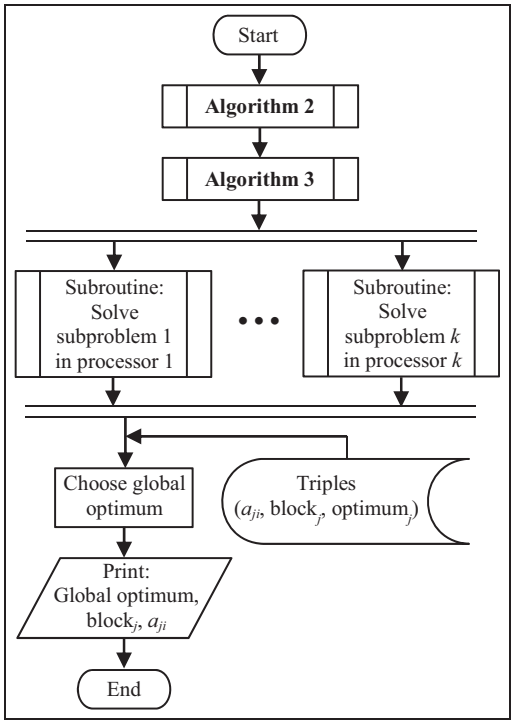

Figure 13. Algorithm 4.

alternative Petri net not shared by the seed. The added link transitions, which correspond to the alternative Petri net a_{ji} should be associated to the choice variable a_{ji} as a guard function.

Step 4. Apply the reduction and simplification rules to the AAPN R_i^A. See Latorre et al.[4] for more details. □

Algorithm 4 describes the complete distributed optimization algorithm, based on a genetic algorithm, to guide the search in the solution spaces of every subproblem. In the implementation of the algorithm, the subproblems of size $m > 1$ are based on an AAPN according to definition 10. Algorithm 4 is shown in Figure 13.

Algorithm 4

Step 1. Apply algorithm 2 in the main processor. Notice that the last ''end'' should be changed to ''return''.

Step 2. Apply algorithm 3 in the main processor or apply a variant of algorithm 3 to be executed in a distributed way. Notice that the last ''end'' should be changed to ''return''.

Step 3. Solve optimization subproblem j in processor j, for $1 \leq j \leq k$.

Step 4. Compare the fitness (value of the objective or cost function) of the k fittest solutions calculated in k processors and choose the best one.

Step 5. Print the information required for the human decision-maker to solve the decision problem based on the original undefined DES: global optimum, block of the partition of S_R that led to the solution, and choice variable, informing on the alternative structural configuration associated to the best solution found. Notice that these data collect the information required to specify values for the degrees of freedom of the undefined DES. □

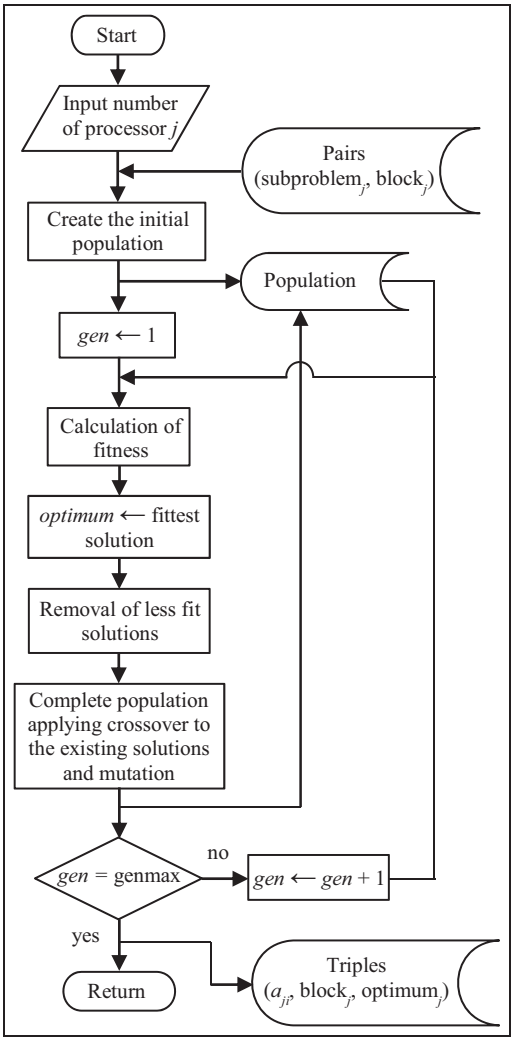

Figure 14. Subroutine for algorithm 4.

The subroutine for algorithm 4 allows solving an optimization subproblem based on an AAPN in a single processor. Its flowchart has been represented in Figure 14.

Subroutine for algorithm 4

Step 1. Select the initial population of feasible solutions for the genetic algorithm.

Step 2. Calculate the fitness of every solution.

Step 3. Store the fittest solution.

Step 4. Remove the less fit solutions.

Step 5. Complete the population with the offspring of the remaining solutions by the application of the crossover and mutation operations.

Step 6. If the number of generations that have been calculated has not reached the expected value, then go to step 2.
Step 7. Return the optimum value, with the block number and the choice variable a_{ji} to be compared with the others returned by the rest of the processors. □

Algorithm 4 and its subroutine should be modified slightly to be adapted to subproblems based on sets of alternative Petri nets instead of a single AAPN.

Algorithm 4(b)

Step 1. Apply algorithm 1 in the main processor. Notice that the last ''end'' should be changed to ''return''.

As it can be seen, the partition of the set of alternative Petri nets S_R, developed as the model of the original undefined DES, should be performed to profit from this specific approach. Algorithm 4 would eventually work with this approach, but its performance is expected to be lower than algorithm 4(b) for the statement of the subproblems according to definition 9 instead of definition 10.

The subroutine for algorithm 4(b) would also be different from the one for algorithm 4. In particular, it should perform a sequential solution of all the subproblems that correspond to definition 7 and the size of the block associated with the current processor. This sequential solving process is associated with the paradigm of ''divide and conquer'', which states that in every processor, as many subproblems (according to definition 7) as alternative Petri nets are associated with the corresponding block of the partition.

As a consequence a repetitive control structure should be inserted in the subroutine:

Step 6b. If there are unsolved subproblems, go to step 1.

Eventually, the result of the subroutine for algorithm 4(b) will be a set of m best solutions arising from the optimization of the m alternative Petri nets associated to the subproblem of size m that corresponds to the current processor. For this reason an additional step is necessary to finish the task of every processor:

Step 6c. Compare the fitness (value of the objective or cost function) of the fittest solutions and choose the best one.

The subroutine for algorithm 4(b), including the mentioned modifications from algorithm 4 has been presented in Figure 15.

Once the different algorithms for solving an optimization problem based on an undefined Petri net according to two distributed approaches have been presented, a comparison of their performance will be described in the following section. For this purpose, a case study is presented.

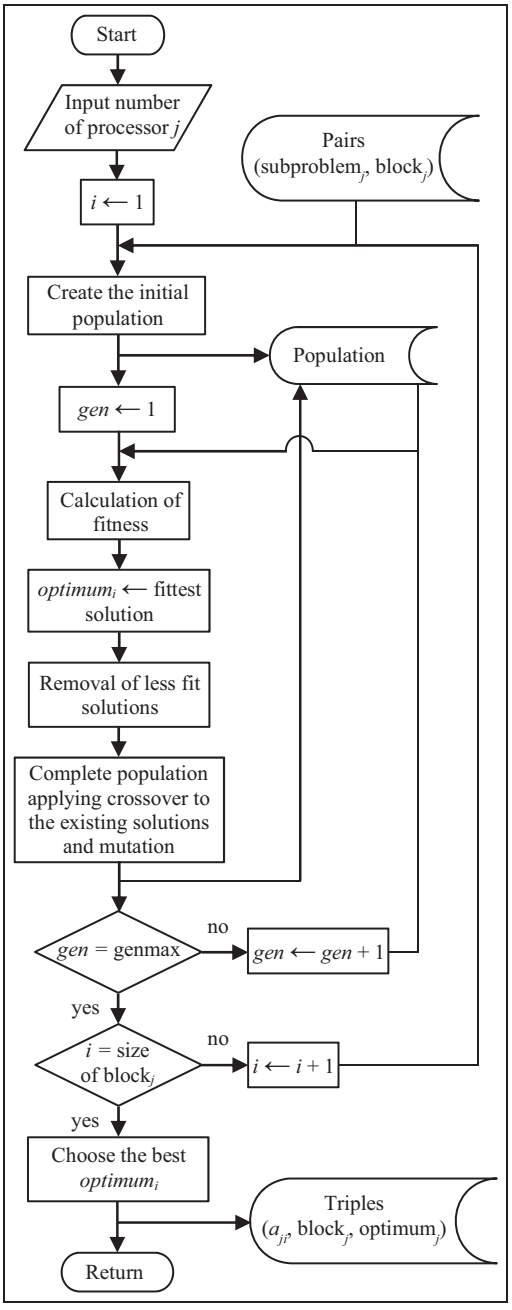

Figure 15. Subroutine for algorithm 4(b).

5. Comparison of the solving methodologies

5.1. Introduction

In this section, a comparison between different methodologies to solve a decision problem related to an undefined DES, modeled by Petri nets and with alternative structural configurations, is presented. Some of the considered methodologies are implemented on a single processor and some of them are distributed to a set of k available processors. Furthermore, the alternative structural configurations of

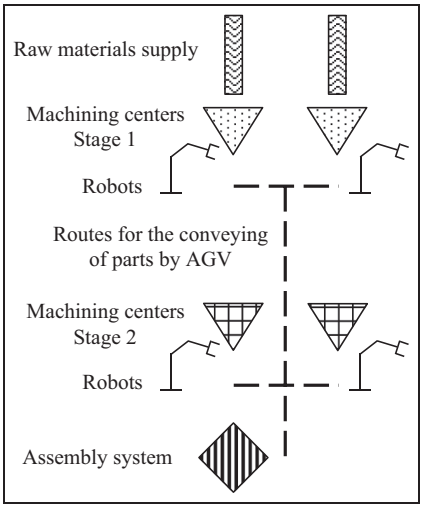

Figure 16. Layout of the flexible manufacturing system of example 3.

the undefined DES are modeled in some of the approaches as a set of alternative Petri nets, as an AAPN, or as a combination of these two formalisms.

The aim of this multiple comparison is to obtain information on the most efficient approach, in terms of required computer resources, among the different modeling strategies and for a given number of available processors. In order to afford this comparison, a case-study is developed with the aim of comparing real measures of computer time needed by the different approaches.

5.2. Case-study: the design of a production facility

In example 3, the benchmark presented by Zhou and Venkatesh[32] and extended by Latorre et al.[44] is solved by means of different approaches, including two distributed algorithms. The results are compared in Table 2 and commented afterwards.

This case-study presents a decision-making problem for the operation of a manufacturing facility, whose layout has been depicted in Figure 16. Several degrees of freedom in the system require making different choices:

(1) choice of the best manufacturing strategy: diverse implementations of pure push and pull paradigms can be chosen, as well as combinations of them in different stages of the production system. The Petri net model of every one of these possibilities constitutes an alternative structural configuration, since the incidence matrix is different in each case;
(2) choice of the production and conveying lot size: the manufacturing system is composed of independent machining centers. The conveying of semi-finished parts is performed by means of automatic guided vehicles (AGVs). Analogously to the previous choice, every lot size is modeled by Petri nets with different incidence matrices; hence, they constitute alternative structural configurations;
(3) choice of the layout of the route or routes of the AGV: every route corresponds to a different alternative structural configuration; and
(4) choice of the number of AGVs and their assignment to the different routes: in this case, the diverse possibilities can be modeled by means of the same Petri net model with different initial markings; hence, they do not lead to alternative structural configurations.

The different structural configurations can be modeled by means of a set of 24 alternative Petri nets, a single AAPN, or a combination of these two formalisms among other possibilities.[42]

The optimization methodology used to solve the version of the case-study presented in the work of Latorre et al.[44] is based on a metaheuristic-guided search in the solution space and the calculation of the objective function by means of the simulation of the behavior of the Petri net model. The chosen metaheuristic is a genetic algorithm.[4]

In order to perform the fairest comparison among the different methodologies, the same genetic algorithm with the same parameter adjustment has been applied to all of them. In particular, a population of 50 solutions for generation has been implemented and a constant value of 15 generations has been considered as criterion to finish the application of the genetic algorithm.

5.3. Estimation of the required computer time

Under the mentioned conditions, a rough estimation of the computing time required by the different methodologies can be performed. Every optimization algorithm is based on multiple evaluations of the objective function, one for every tested solution. Furthermore, every evaluation of the objective function requires the simulation of the evolution of the Petri net model for a certain period of simulated time.

The simulation of the evolution of a Petri net model requires the iterative evaluation of the state equation for the calculation of the sequence of states reached by the model in its evolution. The computing effort of every evaluation of equation (1) is proportional of the size of the incidence matrix, as can be seen in equation (1). As a consequence, a comparison of the computing time required by the optimization processes based on a single Petri net (alternative Petri net or AAPN) can be roughly afforded by the comparison of the sizes of the incidence matrices of every Petri net model.

The size of the incidence matrices of the 24 alternative Petri nets is around 78×58. There is a variation of one or two rows and columns regarding the alternative Petri net. On the other hand, the size of the AAPN is 99×125.

Table 3. Rates of computing time required by different applied methodologies.

t_{column}/t_{row}	(a)	(b)	(c)	(d)	(e)
(a)	1	0.098	0.048	0.096	0.183
(b)	10.22	1	0.494	0.982	1.866
(c)	20.674	2.023	1	1.986	3.775
(d)	10.411	1.019	0.504	1	1.901
(e)	5.477	0.536	0.265	0.526	1

The relative size of the incidence matrix of the AAPN with respect to the incidence matrix of a single alternative Petri net is as follows

$$\approx \frac{99.125}{78.58} = \frac{12375}{4350} = 2.84 \quad (8)$$

As a consequence it is expected that an optimization process associated to a subproblem based on an alternative Petri net may be around 2.84 times faster than the optimization process based on the AAPN.

5.4. Description of the tested approaches

The previous theoretical estimation may be compared to a real measure of the computational effort required by the optimization processes.

A set of tests has been performed in identical computers. The tests that have been developed are the following:

(a) optimization problem based on a set of 24 alternative Petri nets, solved in the same processor: this statement of the problem has been described in definition 6, ''optimization problem based on a set of *n* alternative Petri nets'';
(b) optimization problem based on an AAPN, solved in a single processor: this statement of the problem has been described in definition 8, ''optimization problem based on an AAPN'';
(c) set of optimization subproblems of size 1 based on a single alternative Petri net: every subproblem is solved in a different processor. This statement of the subproblems has been described in definition 7, ''optimization subproblem based on an alternative Petri net'';
(d) set of $k = 12$ optimization subproblems based on subsets of alternative Petri nets: k is also the number of processors. This statement of the subproblems has been described in definition 9, ''optimization subproblem of size m based on a subset of alternative Petri nets''; and
(e) set of $k = 6$ optimization subproblems based on subsets of alternative Petri nets: k is also the number of processors. This statement of the subproblems has been described in definition 9, ''optimization subproblem of size m based on a subset of alternative Petri nets''.

5.5. Quantitative results of the tests and comparison

A comparison of the results can be seen in Table 3, where the pairwise rates of the computer times required by the different tested methodologies have been written.

In every position of the table, the speed of the solving methodology represented in the row with respect to the solving methodology indicated in the column has been indicated. For example, the value written in the position (3, 2) of Table 3, which can be found in the third row and second column, has been calculated by dividing the computer time required by methodology (b) by the time that methodology (c) requires. The value associated to this position is 2.023, meaning that methodology (c) (row) is 2.023 times faster than methodology (b) (column).

5.6. Comparison between the approaches applied on a single processor

In order to evaluate the quality of equation (8) with respect to the computer time required to perform an optimization based on a single alternative Petri net and an AAPN it is possible to divide the average of the 24 times obtained in (c) with the time measured in (b). The result, which is the speed of the approach based on a single AAPN versus the set of alternative Petri nets, is 2.35 versus the theoretical 2.84. In fact, if a comparison of every one of the computer times required for the different subproblems of size 1 with the computer effort required by the AAPN is made, the interval is obtained as follows: [2.02, 2.64].

The difference between the theoretical and practical results may be due to the fact that since the number of tested solutions and evaluated generations is the same for every problem or subproblem, the amount of simulations for tested solutions and the number of evaluations of equation (1) for the simulation may be different in every case. In the theoretical estimation, all these figures have been taken as the same for all the cases in order to simplify the calculations.

5.7. The fastest algorithm

Interesting deductions can be made from the data represented in Table 3. On the one hand the slowest approach consists with the strategy of ''divide and conquer'' implemented on a single processor (a), which is 10 times slower

when compared to the other single-processor approach, based on the AAPN.

From Table 3, it is clear that the fastest algorithm of all the methodologies is the one implemented in the highest number of processors. In this case, it has been used in 24 processors; as many processors as alternative Petri nets are needed to model the original undefined DES (c). This process is about 20 times faster than (a) and only twice as fast as (b).

This last result is surprisingly low considering that in the AAPN-based case, (b), a single processor is used versus 24 processors required in case (c). As it can be seen, methodology (b) is a very efficient option for a single processor. As a consequence, the use of AAPN models in combination with a larger number of processors might lead to very efficient methodologies.

Of course, it has to be considered that using techniques for distributing the solution of an optimization subproblem of size 1 into more than one processor, in addition to the mentioned methodologies, the computer time of the approach (c) could be improved. Nevertheless, in this paper, these additional techniques, such as those presented in section 1.2, have not been taken into account, because the objective of this paper is to analyze the efficiency of the specific methodologies mentioned in section 5.4.

5.8. Results regarding a limited number of available processors

Other interesting results arise when the optimization algorithms are implemented in a number of processors smaller than n, the number of alternative Petri nets in S_R or the number of subproblems of size 1. In this case-study, $n = 24$.

In order to implement algorithm (c) on k processors, such that $k < n$, cases (d) and (e) have been solved, where 12 and 6 available processors are used respectively.

Both cases, (d) and (e), are more general than it might be supposed at first glance. In both cases every processor j will solve sequentially a set of m_j subproblems of size 1. The value of m_j will depend on the size of $S_R = n = 24$, and the number of available processors k. See algorithm 1.

Case (d) is based on the availability of 12 processors. For this reason, the number of assigned subproblems to every processor will be $n / k = 24 / 12 = 2$. Any case in this example, where the number of available processors k verifies that $n > k \geq n / 2$, is expected to require similar computer time for solving the complete optimization problem. Actually, the time needed for solving the complete problem would be imposed by the processor requiring more time.

In the cases that range from $n / 2 = 12$ to $n - 1 = 23$ available processors, there is at least one processor, which should solve sequentially two optimization subproblems of size 1. Due to the fact that every subproblem requires the same time to be solved, all the mentioned cases, which assign at least two subproblems to a single processor, will require a similar amount of time to solve the complete optimization problem. It is necessary to clarify that the time required for any of the $n = 24$ subproblems to be solved is roughly the same, since the dimensions of the incidence matrix of every alternative Petri net is almost the same. In effect, the speed of (c) with respect to (d), obtained from the tests, is 1.986.

Furthermore, an indication of the efficiency of solving the problem with an AAPN model and a single processor can be obtained by comparing the computer time required in (b) with the tests performed in (d). In fact, the tests performed by the authors have determined the average rate time(b)/time(d) in the interval [1.01, 1.31] as 1.019.

On the other hand, case (e) is based on the availability of six processors. For this reason, the number of assigned subproblems to every processor will be $n / k = 24 / 6 = 4$. Any case in this example, where the number of available processors k verifies that $8 = n / 3 > k \geq n / 4 = 6$, is expected to require similar computer time for solving the complete optimization problem.

In these cases, since all the $n = 24$ subproblems of size 1 require approximately the same amount of time to be solved, it is expected that quadruple the computer time will be required for implementing the sequential solution of four subproblems of size 1. According to this consideration, the optimization time required for algorithm (e), distributed into a number of computers belonging to $[n / 4, n / 3 - 1] = [6, 7]$ is approximately equal.

As expected, the rate of time(e)/time(c) that has been measured is 3.775. Furthermore, the tests performed by the authors in the application of case (e) have led to a measured computing time 1.86 times higher in the ''divide and conquer'' approach (e) than in the single-processor approach based on the AAPN approach (b). This value of 1.86 is the speed of (b) with respect to (e).

As a conclusion, it is possible to say that in this case-study the AAPN approach (b) implemented in a single processor reaches roughly the computer time required by a distributed algorithm under the approach of ''divide and conquer'' (d) for k processors such that $n > k \geq n / 2$. In fact, the single-processor approach is less efficient than the ''divide and conquer'' approach in this interval. Furthermore, the single-processor AAPN-based method (b) improves the results for $k < n / 2$ processors, for example (e).

It is possible to deduce that the AAPN approach is very efficient at solving certain types of optimizations. According to the different applications of this methodology performed by the authors, the more shared subnets and the larger they are the better the results offer by the AAPN-based methodology are compared to the approach of ''divide and conquer''.

As a consequence of the previous conclusions, the interest of performing tests of distributed computing associated

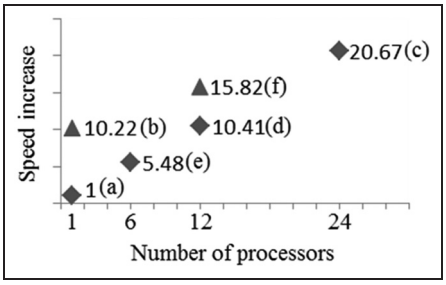

Figure 17. Speed increase of the different approaches.

to subproblems of size $m > 1$ based on an AAPN instead of sets of alternative Petri nets has been justified, as in cases (c), (d), and (e).

In order to analyze the efficiency of a distributed approach based on AAPN models the following test has been performed:

(f) set of $k = 12$ optimization subproblems of size 2 based on an AAPN: k is also the number of processors. This statement of the subproblems has been described in definition 10, ''optimization subproblem of size m based on an AAPN''.

The test of this case-study performed on a set of $n / 2 = 12$ computers has led to a rate of computing time time(d)/time(f) measured as 1.52. This value means that for this case-study the distributed approach based on an AAPN for $k = 12$ processors presents a speed increase of 1.52 versus the distributed approach based on sets of alternative Petri nets for the same number of processors. Moreover, for the reasons mentioned in the previous paragraphs, it is likely that the speed increase is roughly the same for a number of processors belonging to the interval [12, 23].

As a summary of the results described in the present section, Figure 17 is shown. The x-axis represents the number of processors in which the case-study is solved, from 1 to 24. The y-axis represents the speed increase of every methodology studied, from (a) to (f), with respect to the single-processor approach based on a set of alternative Petri nets (a). Notice that approach (a) has a speed increase of 1. In Figure 17 both approaches based on AAPN models, (b) and (f), have been represented by triangles, whereas the others, based on a set of alternative Petri nets, have been depicted by means of a diamond. To the right of the mentioned symbol, the speed increase and the approach ranging from (a) to (f) has been indicated.

It has to be noticed that in all the studied tests, the optimum found was associated to an objective function that takes similar values; hence, the quality of the obtained solutions are very similar. □

6. Conclusions and further research

In this paper, the solution of decision problems based on undefined DESs, which include a set of alternative structural configurations, has been discussed. Two modeling formalisms based on the paradigm of the Petri nets have been presented. The first one, based on a classic approach of ''divide and conquer'', is a set of alternative Petri nets, where each alternative Petri net models a single alternative structural configuration. The second one is based on the compact formalism of the AAPN, which can profit from the existence of shared subnets in the different alternatives. This property allows the AAPN to remove the redundant information contained in the set of alternative Petri nets.

It has been shown that the original decision problem can be formalized into different statements of the same optimization problem. Some of the statements are to be implemented on a single processor, while others profit from the use of a higher number of processors.

A problem of size m, associated to a set of m alternative Petri nets, can be partitioned into several subproblems associated to k different processors, where $k \leq m$. The number of feasible partitions has been calculated by means of the Stirling numbers of the second kind.

In fact, this paper has shown, via the analysis of a case-study, that the fastest statement of the optimization problem consists in the use of $k = m$ different processors to solve optimization problems of size 1 based on a single alternative Petri net.

If the number of available processors is larger than m, it might be possible to accelerate the optimization process. Nevertheless, further research is needed in this field.

On the other hand, if the number of available processors is smaller than m, then this methodology of ''divide and conquer'' is not necessarily the most efficient one.

In fact, in the interval $k \in [n / 3 - 1, n / 4]$ of processors, the single-processor AAPN-based statement of the problem has shown to be the most efficient. In the interval $k \in [n - 1, n / 2]$ both statements of the problem require a similar computer time to finish; however, the approach based on a set of alternative Petri nets is slightly better than the single-processor AAPN-based approach.

Furthermore, the combination of a distributed algorithm with the construction of an AAPN for the sets of alternative Petri nets associated to a subproblem of size $m > 1$ is a very promising methodology as it has been shown in the case-study. In fact, this methodology has outperformed the best analyzed strategy for the case $k < m$.

This promising methodology has a significant potential for the case with a high number n alternative structural configurations. A large value for n can be obtained easily in the design of a DES, due to combinatorial reasons, which might even produce a combinatorial explosion. In such a case, the number of available processors, k, is likely to be smaller than the number of alternative Petri nets, which is the situation studied in more detail in this paper.

As next steps in the development of this research, it is convenient to apply it to a broader range of real cases in

order to understand better the advantages and drawbacks of the methodology, as well as to predict its performance for a given application.

Funding

This paper has been partially supported by the project of the University of La Rioja (UR) and Banco Santander (grant number API12-11) ''Sustainable production and productivity in industrial processes: integration of energy efficiency and environmental impact in the production model for integrated simulation and optimization''.

References

1. Cassandras CG and Lafortune S. *Introduction to discrete event systems*. New York: Springer, 2008.
2. Silva M and Teruel E. Petri nets for the design and operations of manufacturing systems. *Euro J Control* 1997; 3: 82–199.
3. Zimmermann A, Rodríguez D and Silva M. A two phase optimisation method for Petri net models of manufacturing systems. *J Intell Manufact* 2001; 12: 409–420.
4. Latorre JI, Jiménez E and Pérez M. A genetic algorithm and Petri nets approach for decision problems stated on discrete event systems. In: *12th international conference on computer modelling and simulation (UKSIM2010)*, Cambridge, UK, 24–26 March 2010, pp. 86–91.
5. Narciso M, Piera MA and Guasch A. A methodology for solving logistic optimization problems through simulation. *SIMULATION* 2010; 86: 369–389.
6. Mustafee N, Taylor SJE, Katsaliaki K, et al. Facilitating the analysis of a UK National Blood Service supply chain using distributed simulation. *SIMULATION* 2009; 85(2): 113–128.
7. Tang Y, Perumalla KS, Fujimoto RM, et al. Optimistic parallel discrete event simulations of physical systems using reverse computation. In: *Workshop on principles of advanced and distributed simulation (PADS'05)*, Washington, DC, USA, 1–3 June 2005.
8. Vitali R, Pellegrini A and Cerasuolo G. Cache-aware memory manager for optimistic simulations. In: *SIMUTools 2012*, Sirmione, Italy, 19–23 March 2012.
9. Ferscha A. Parallel and distributed simulation of discrete event systems. In: Zomaya AY (ed) *Parallel and distributed computing handbook*, New York: McGraw Hill, 1995, pp. 1003–1041.
10. Page EH, Litwin L, McMahon MT, et al. Goal-directed grid-enabled computing for legacy simulations. In: *2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID '12)*, Ottawa, Canada, 13–16 May 2012, pp. 873–879.
11. Andjelković B, Litovski VB and Zerbe V. Grid-enabled parallel simulation based on parallel equation formulation. *ETRI J* 2010; 32: 555–565.
12. Evans NS, GauthierDickey C, Grothoff C, et al. Simplifying parallel and distributed simulation with the DUP system. In: *43rd annual simulation symposium (ANSS'10)*, Orlando, Florida, USA, 12–15 April 2010, pp. 208–215.
13. Murshed M, Buyya R and Abramson D. GridSim: a toolkit for the modeling and simulation of global grids. Technical report, Monash-CSSE 2001/102, Monash University, Australia, 2001.
14. Wozniak JM, Brenner P, Thain D, et al. Generosity and gluttony in GEMS: grid enabled molecular simulations. In: *14th IEEE international symposium on high performance distributed computing (HPDC 2005)*, Research Triangle Park, North Carolina, USA; July 24–27, 2005.
15. Wainer G, Liu Q, Chazal J, et al. Performance analysis of web-based distributed simulation in DCD++: a case study across the Atlantic Ocean. In: *2008 spring simulation multiconference (SpringSim)*, April 2008, pp. 413–420.
16. Timm IJ and Pawlaszczyk D. Large scale multiagent simulation on the grid. In: *5th IEEE international symposium on cluster, computing and the grid (CCGRID '05)*, Cardiff, UK, 9–12 May 2005, vol. 01, pp. 334–341.
17. D'Angelo G and Bracuto M. Distributed simulation of large-scale and detailed models. *Int J Sim Process Modelling* 2009; 5: 120–131.
18. Martin JM. *Parallel discrete event simulation of large scale wireless ad-hoc networks*. PhD Dissertation, University of California, Los Angeles, USA, 2002.
19. Fujimoto RM, Perumalla K, Park A, et al. Large-scale network simulation: how big? how fast? In: *11TH IEEE/ACM international symposium on modeling, analysis and simulation of computer telecommunications systems (MASCOTS'03)*, Orlando, Florida, USA, 2003, pp. 116–123.
20. Theodoropoulos G, Zhang Y, Chen D, et al. Large scale distributed simulation on the grid. In: *CCGRID 2006 workshop: international workshop on distributed simulation on the grid*, Singapore, 16–19 May 2006.
21. Aydt H, Turner SJ, Cai W, et al. Symbiotic simulation systems: an extended definition motivated by symbiosis. In: *22nd workshop on principles of advanced and distributed simulation (PADS '08)*, Rome, Italy, 3–6 June 2008, pp. 109–116.
22. Chiola G and Ferscha A. Distributed simulation of Petri nets. *IEEE J Parallel Distributed Tech* 1993; 1: 33–50.
23. Hulaas J. An evolutive distributed algebraic Petri nets simulator. In: *10th European simulation multiconference ESM'96*, Budapest, Hungary, 1996, pp. 348–352.
24. Holvoet T and Verbaeten P. Using agents for simulating and implementing Petri nets. In: *11th workshop on parallel and distributed simulation (PADS'97)*, Austria, 1997, pp. 134–137.
25. Kuo CH. Development of distributed agent-oriented Petri net simulation and control environment for discrete event dynamic systems. In: *IEEE international conference on systems, man and cybemetics (SMC 2004)*, The Hague, The Netherlands, 10–13 October 2004, vol. 5, pp. 5001–5006.
26. Nicol DM and Mao W. Automated parallelization of timed petri-net simulations. *J Parallel Distributed Comp* 1995; 29: 60–74.
27. Knoke M, Kühling F, Zimmermann A, et al. Towards correct distributed simulation of high-level Petri nets with fine-grained partitioning. In: *Lecture notes in computer science, vol. 3358: parallel and distributed processing and applications: second international symposium, ISPA 2004* (eds Cao J, Yang LT, Guo M, et al.), Hong Kong, China, 13–15 December 2004, pp. 64–74. New York/Heidelberg: Springer-Verlag.

28. Zimmermann A, Knoke M and Hommel G. Complete event ordering for time-warp simulation of stochastic discrete event systems. In: *4th symposium on design, analysis and simulation of distributed systems (DASD 2006)*, Huntsville, Alabama, USA, 2–6 April 2006.
29. Haggarty OJ, Knottenbelt WJ and Bradley JT. Distributed response time analysis of GSPN models with MapReduce. *SIMULATION* 2009; 85: 497–509.
30. Yoo T, Cho H and Yücesan E. Web services-based parallel replicated discrete event simulation for large-scale simulation optimization. *SIMULATION* 2009; 85: 461–475.
31. Zuberek WM. Performance study of distributed generation of state spaces using coloured Petri nets. In: *4th workshop and tutorial on practical use of coloured Petri nets and the CPN tools CPN'02* (ed Jensen K), Aarhus, Denmark, 28–30 August 2002, pp. 81–98.
32. Zhou M and Venkatesh K. *Modelling, simulation and control of flexible manufacturing systems. A Petri net approach*. Singapore: WS World Scientific, 1999.
33. Mušič G. Petri net based scheduling approach combining dispatching rules and local search. In: *21st European modelling and simulation symposium (EMSS 09)*, Puerto de la Cruz, Spain, September 2009, vol. 2, pp. 27–32.
34. Piera MA and Mušič G. Coloured Petri net scheduling models: timed state space exploration shortages. *Math Comp Sim* 2011; 82: 428–441.
35. Bai Q, Ren F, Zhang M, et al. Using colored Petri nets to predict future states in agent-based scheduling and planning systems. *J Multiagent Grid Systems - Advances Agent-mediated Auto Neg Archive* 2010; 6; 527–542.
36. Nishi T and Maeno R. Petri net decomposition approach to optimization of route planning problems for AGV systems. *IEEE Trans Auto Sci Engr* 2010; 7: 523–537.
37. Tsinarakis GJ, Tsourveloudis NC and Valavanis KP. Petri net modeling of routing and operation flexibility in production systems. In: *13th Mediterranean conference on control and automation*, Limassol, Cyprus, 27–29 June 2005, pp. 352–357.
38. Silva M. Introducing Petri nets. In: DiCesare F (ed) *Practice of Petri nets in manufacturing*. London: Chapman and Hall, 1993, pp. 1–62.
39. Balbo G and Silva M (eds). *Performance models for discrete event systems with synchronizations: formalisms and analysis techniques*. Saragosse, Spain: Editorial Kronos, 1998.
40. David R and Alla H. *Discrete, continuous and hybrid Petri nets*. Berlin: Springer, 2005.
41. Recalde L, Silva M, Ezpeleta J, et al. Petri nets and manufacturing systems: an examples-driven tour. Lectures on concurrency and Petri nets: advances in Petri nets. In: Desel J, Reisig W and Rozenberg G (eds) *Lecture notes in computer science*. New York: Heidelberg: Springer-Verlag, 2004, vol. 3098, pp. 742–788.
42. Latorre JI, Jiménez E and Pérez M. The optimization problem based on alternatives aggregation Petri nets as models for industrial discrete event systems. *SIMULATION* 2013; 89: 346–361.
43. Latorre-Biel JI, Jiménez-Macías E and Pérez M. The exclusive entities in the formalization of a decision problem based on a discrete event system by means of Petri nets. In: *23rd European modelling and simulation symposium (EMSS 11)*, Rome, Italy; September 2011, pp. 580–586.
44. Latorre JI, Jiménez E, Pérez M, et al. The alternatives aggregation Petri nets as a formalism to design discrete event systems. *Inter J Sim Process Modeling* 2010; 6: 152–164.
45. Rota GC. The number of partitions of a set. *Amer Math Monthly* 1964; 71: 498–504.
46. Bell ET. Exponential numbers. *Amer Math Monthly* 1934; 41: 411–419.
47. Joarder A and Mahmood M. An inductive derivation of stirling numbers of the second kind and their applications in statistics. *J Applied Math Decision Sci* 1997; 1: 151–157.

Author biographies

Juan-Ignacio Latorre is an industrial engineer. He has developed his professional career in industry and in educational institutions. Currently he is an assistant professor in the Department of Mechanical Engineering, Energetics, and Materials of the Public University of Navarre (Spain) and in the Department of Mechanical Engineering of UR. His research interests include factory automation, simulation, and modeling of industrial processes and formal methodologies to solve decision problems based on DESs. His main research activities have been performed with the research Group of Modeling, Simulation, and Optimization of UR, in the field of decision support systems in DES-based on Petri net models.

Emilio Jiménez is a professor at UR in the Electrical Engineering Department, where he leads the research Group of Modeling, Simulation, and Optimization. His main research interests include factory automation, modeling, and simulation of industrial processes and formal methodologies to solve decision problems based on DESs. His main research activities have been performed with the Group of Discrete Event Systems Engineering of the University of Zaragoza, Spain, and the Modeling, Simulation, and Optimization Group of UR, in the field of decision support systems in DESs based on Petri net models.