A Methodology for Solving Logistic Optimization Problems through Simulation

Mercedes Narciso
Miquel A. Piera
Departament de Telecomunicació i Enginyeria de Sistemes,
ETSE (Edifici Q),
Universitat Autònoma de Barcelona,
08193 Bellaterra,
Barcelona, Spain
mercedes.narciso@uab.es
miquelangel.piera@uab.es

Antoni Guasch
Departament d'Enginyeria de Sistemes i Automàtica,
FIB (Campus Nord),
Universitat Politècnica de Catalunya,
08034 Barcelona,
Spain
toni.guasch@upc.edu

In this paper we present a methodological approach designed to automate the decision-making in logistic systems, with deterministic time, by solving optimization problems. The Colored Petri Net (CPN) formalism has been used as a base to develop a methodology that integrates the features of operational research, artificial intelligence and simulation fields. At the same time, it combines the modeling of discrete event systems with simulation, analysis and system optimization, transforming a conceptual model into a simulation model, and a decision problem into a search problem. The use of the CPN formalism has allowed the integration of all of these different research fields into a unique decision support tool.

Keywords: discrete event systems, optimization problems, scheduling, colored Petri nets, decision support systems

1. Introduction

A methodology for solving optimization problems, in the logistics field, should explain how to go from the description of a problem related to a logistic process to its specification in a formal model. This model should represent the behavior of the process, and indicate how it could be solved in order to find and optimize the solution of the problem. With the purpose of improving the process performance, this should be done in a way that ensures that this solution could be used in the decision-making activity.

The complexity for the decision-making in the logistics field appears due to the relations among the activities that should be properly coordinated, as well as the consequences (effects) of the different decisions. An evaluation of all of the possible decision alternatives requires an evaluation of the different states that the system can reach. It is considered to be a NP-hard combinatorial problem that cannot be solved by using polynomial time algorithms.

It is important to emphasize that interest in the solution to this type of problem in the logistics field is related to the economic advantages that would be obtained from a

SIMULATION, Vol. 86, Issue 5-6, May-June 2010 369–389
© 2010 The Society for Modeling and Simulation International
DOI: 10.1177/0037549709356011
Figures 1, 4–9 appear in color online: http://sim.sagepub.com

better coordination of the production operations, as well as, among others, transportation and storage operations. In this context, the important individual efforts that have been made up to now in the areas of Artificial Intelligence (AI), Operational Research (OR) and simulation cannot be considered to be completely satisfactory. There are several promising research modeling methodologies that combine AI, OR and/or simulation methods. Although the results obtained are quite promising for some industrial and logistic problems, these problems are still considered nowadays a frontier research area, in which there is a no generic well-accepted solution.

As the logistic problems are usually NP-hard, no standard solution techniques are available and in many cases feasibility problems rather than optimization problems are faced. The discrete optimization approaches used most often to solve such problems are deterministic methods, such as mathematical programming including Mixed Integer Linear Programming (MILP) and Mixed Integer Nonlinear Programming (MINLP), or logic-based optimization including Generalized Disjunctive Programming (GDP) and Constraint Programming (CP). The application of these methods has successfully solved many logistic and industrial problems [1–5].

Although using these deterministic methods, problems can be solved quite efficiently, sometimes do not solve certain practical problems in reasonable time, and require a deep knowledge on computational causality (i.e. MILP models can consist up to hundreds of thousands of constraints and variables for some industrial problems) to deal with appropriate model reformulations, generate new algorithms to improve the convergence to the optimal solution, and/or combine with other methods, including heuristics and metaheuristics [6–9].

Thus, from the computational side, the inherent complexity of logistic problems can easily exceed hardware and algorithmic capabilities and, from the modeling side, model formulation and model maintenance can become a hard task when using these optimization methods. Some modeling approaches have been developed to alleviate the problem formulation. One of most relevant contributions in the industrial field for discrete time models is the State-Task Network (STN) representation [10], which allows the unambiguous representation of processing networks of any complexity.

The STN represents physical processes as recipes, by using only two types of nodes: state nodes that correspond to feeds, intermediates and final products; and task nodes that represent processing operations. In general, it is assumed that each process unit can perform several of the tasks in the STN network. The STN representation is translated and solved as a MILP model. Another representation, the Resource-Task Network (RTN), proposed in [11], leads to a more compact model than the STN. It is interesting to note that both the STN and RTN models can handle networks with arbitrary topology, flow equations for the mass balances, all types of transfers and resource constraints, and are based on a discrete or continuous time representation [2].

In this paper we propose an alternative to represent Discrete Event Systems (DESs) by using the Colored Petri Net (CPN) formalism [12, 13], which is ideal to capture the cause-and-effect relationship between different system states, formalizing in this way the knowledge of the system. In fact, the Petri Net (PN) formalism is famous for its solid theoretical base and clear syntax, which is also used in the optimization and control problems of manufacturing systems [14, 15]. However, PN lack primitives to support information flow, which is critical in any logistic or production decision-making activity. CPN can be considered as an extension of the original PN formulation to support information aspects.

A comparative study among deterministic methods and optimization analysis tools based on the CPN formalism is not easy and it is beyond the scope of this paper. Some comparative results between a CPN representation and MILP can be found in [16, 17], which illustrate a comparison between MILP and PN. Actually, it is well accepted that the CPN formalism is very useful to explicitly represent the states and system events, using an intuitive graphical representation with a very precise and unambiguous semantic that supports the analysis of concurrency, synchronism and causal dependency between subsystems, which are very typical dynamics that appear in logistic systems. In [12, 18–20], the main advantages of using PN and CPN to model industrial systems are described. According to the research groups involved in the CPN area, it is expected that powerful CPN-based tools will be available for industrial use in the near future [21].

Regarding the STN representation, there exists graphical similarities between STN and PN/CPN (i.e. both formalisms use only two type of nodes), but the STN describes processes as a network of tasks and states, while CPN describes processes as a set of interrelated events. A comparative between STN and PN (extensible to CPN) is shown in [22]. It is emphasized that a fundamental difference is observed in the time domain representation, since CPN works in the discrete event time domain.

Although the CPN formalism supports stochastic time models by using Stochastic Colored Petri Nets [23, 24], the proposed methodological approach only supports the analysis of deterministic time models. The methodology presented here attempts to be useful in solving optimization problems for logistic systems that could be abstracted as DESs. It proposes the integration of evaluation methods (simulation) with search methods and optimization (AI, OR). This integration would take place through the construction of state spaces of the CPN models [12, 13, 25], all in order to satisfy the requirements for the solution to this type of industrial problem.

In contrast to other research alternatives that use the CPN as a modeling formalism, which only treat problems within very specific areas [25–29], the developed methodology can be applied to solve DES optimization problems in general as well as logistic decision problems in particular. It also covers different phases such as a description of the problem, the system model, efficient representation of knowledge, construction and simulation of the decision model, and analysis of the simulation results. A computational tool is available to aid the decision-making and acts as support to this methodology. Although this tool brings important benefits to the solution of optimization problems, thanks to the implementation of analysis methods, the reduction of the state space (the set of all states that are achievable from the initial state), and methods used to reduce computational time, unfortunately global optimality cannot be guaranteed except for those systems with an upper bound in the number of states to be evaluated.

In Section 2 of this paper we present the CPN formalism, and in Section 3 we describe the requirements on which the development of the methodology has been based. In Sections 4 and 5 we describe the methodological proposal as well as the methodological steps that have been developed. Section 6 illustrates the application of the methodology in order to resolve a reference optimization problem (benchmarking), representative of a real-world logistic system. Finally, in Section 7 we present the conclusions obtained on the development of the methodology.

2. Colored Petri Net Formalism

The CPN formalism is well known by the AI, OR and simulation community for its capability to represent, simulate and analyze DESs. CPN has been chosen as the modeling formalism due to its ability to describe the complete structure of a system together with its behavior and the information about the system state through the use of a graphical representation. A CPN is an extension of the PN formalism, and can be defined as a bipartite directed graph describing the structure of a DES, while the dynamics of the system are described by the simulation of the CPN. The main CPN components are *place nodes, transition nodes, directed arcs, arc expressions, guards, color sets* and *tokens* [12, 13]. Mathematically, a CPN can be defined as a tuple of nine elements:

$$\text{CPN} = (\Sigma, P, T, A, N, C, G, E, I)$$

where:

- $\Sigma = \{C_1, C_2, \ldots, C_{nc}\}$ is a finite set and not empty of color sets, *nc* is the total number of color sets in the CPN model; this allows the specification of attributes that must be defined for every entity type (token) that needs to be modeled;
- $P = \{P_1, P_2, P_3 \ldots\ldots P_{np}\}$ is a finite set of place nodes that permits the system's state specification, *np* is the total number of place nodes in CPN model; each place node contains tokens and is defined by the color of each token;
- $T = \{T_1, T_2, T_3 \ldots T_{nt}\}$ is a finite set of transition nodes, *nt* is the total number of transitions nodes in the CPN model;
- $A = \{A_1, A_2 \ldots A_{na}\}$ is a finite set of arcs such that $P \cap T = P \cap A = T \cap A = \emptyset$, *na* is the total number of arcs in the CPN model;
- N is a node function that allows the association of each arc with its node terminals (origin and destination nodes); the nodes must be of different types, i.e. if a node is a transition, the connecting node must be a place and vice versa

$$\begin{aligned} N : A \to (P \times T) \cup (T \times P) | \forall A_i \in A \exists! P_j \\ \in P \wedge \exists! T_k \in T : [N(A_i) \\ = (P_j, T_k) \vee N(A_i) = (T_k, P_j)]; \\ i = 1, 2, \ldots na;\; j = 1, 2, \ldots np; \\ k = 1, 2, \ldots nt \end{aligned}$$

- C is a color function that allows the specification of the entity type (colored token) that can be stored in every place node

$$\begin{aligned} C : P \to \Sigma | \forall P_j \in P \exists! C_q \in \Sigma : \left[C(P_j) = C_q\right]; \\ j = 1, 2, \ldots np;\; q = 1, 2, \ldots nc \end{aligned}$$

- G is a guard function that is associated with a transition node; it enables or disables an event associated with the transition in function of the color value of the entity attributes to be processed

$$\begin{aligned} G : T \to Boolean | \forall T_k \in T : [type(G(T_k)) \\ = Boolean \wedge type(variables(G(T_k)) \\ \subseteq \Sigma];\; k = 1, 2, \ldots nt \end{aligned}$$

- E is an arc expression function; P_j can represent an input place (arc A_i from place node to transition) or an output place (arc A_i from transition to place node); the arc expression of input places indicates the subset of tokens that can be used to *enable* a transition; an enabled transition can be *fired*, using the arc expressions of output places to compute the new color values of the generated tokens

$$\begin{aligned} E : A \to C(P_j) | \forall A_i \in A : [tipo(E(A_i)) \\ = C(P_j) \wedge tipo(variables(E(A_i))) \\ \subseteq \Sigma];\; i = 1, 2, \ldots na;\; j = 1, 2, \ldots np \end{aligned}$$

- I is an initialization function that specifies color values of the initial tokens stored in each place; this is referred to as the initial state of the CPN model

$$I : P \to C(P_j) | \forall P_j \in P : [tipo(I(P_j)) = C(P_j)];\; j = 1, 2, \ldots np$$

The place nodes are represented graphically by circles (or ellipses) and can be used to describe the system components (resources and entities). In general, they describe the state of system components. The transition nodes are represented graphically by rectangles and can be used to describe events, actions or activities that appear in the dynamics of a system. The directed arcs are represented graphically by arrows, and are used to connect a place with a transition, or a transition with a place. The tokens associated with the place nodes are used to model the state of the system components, for example, the number and type of entities (parts, resources, people, etc.), or the state of resources (working, free, idle, etc.). In a CPN model, each token has a data type attached called the *token color*. For a given place node, all tokens must have colors (*colored tokens*) that belong to the specified type. This type is called a color set of the place node, and is indicated by the name of the color set on one side of the place node. Color combinations determine the possible values for the tokens. The initial color tokens are specified by an underlined expression next to the place node, with the following information:

$$n'(c_1, c_2, c_3, \ldots, c_r, \ldots, c_{nr})$$

in which n represents the number of tokens with the color values described in parentheses, c_r represents the value of a color, and nr is the number of attributes of the color.

When the color values of tokens are not identical for all elements of the same node, the '+' is used to specify the colors values of each token:

$$n'_1(c_{11}, c_{12}, c_{13}, \ldots, c_{1r}, \ldots, c_{1nr}) + n'_2(c_{21}, c_{22}, c_{23}, \ldots, c_{2r}, \ldots, c_{2nr}).$$

The arc expressions describe the conditions for state changes occurring in the system. Tokens are inspected by the input arc expressions to a certain transition, in order to determine whether the transition is *enabled* or not. When a transition is enabled it can be fired. The effect of firing a transition is that some tokens are removed from place nodes connected at the input of the transition and added to place nodes connected at the output of the transition. The number and type of tokens removed/added is specified by the arc expressions associated with the input/output transition arcs. Guards expressions can be considered logical conditions attached to each transition, which also determine whether a certain transition can be fired or not.

Guards are formalized graphically in brackets '[ ]' located next to the transition. The *marking* represents the minimum information needed to predict which potential events may occur (i.e. the state of the system). Marking is specified by the number of tokens in each place node and the color values of each token.

Figure 1 illustrates the graphical components of a CPN model. A marking M_s represents a state of the system, which is described by specifying the colored tokens in each place node of the CPN. The *initial marking* (initial state of the system) is represented by marking M_0.

3. Requirements of the Methodology

The development of the methodology is based on three requirements considered essential to solving an optimization problem in the logistics field. These conditions have been taken into consideration because they summarize the process for solving this kind of problem. They begin by modeling the problem (modeling requirement), going on to analyze the different decision alternatives that can be used in problem solving (analysis requirement), and finish by selecting the best alternative decision: the one that optimizes the solution to the problem (optimization requirement).

3.1 Modeling Requirement

In order to develop the methodology, logistic systems that can be represented as a DES have been considered. The main behavioral feature of a DES is that the system does not change its state until an event occurs (i.e. an event generates an instantaneous change in the state of the system). Time advances from event to event and the latter take place chronologically, but not necessarily at regular time intervals of time. The complexity of decision-making in such systems is that any decision can block, freeze, delay, enable/disable future events (upstream, downstream) even if not directly related to the event associated with the decision.

In addition, it must be taken into account that the aspects of concurrency, synchronization and parallelism in a DES are critical to decision-making. For example, consider the interaction of processes in a manufacturing system: its dynamic nature and the need for a proper coordination of machines, tools, equipment and operators are critical aspects for improving production performance. When production operations use the same resources (finite) such as machines, stocks, tools, the competition for resources transforms the flow of information into a vital function for dealing with a proper system performance. A scheduling policy must then deal with the timely allocation of manufacturing operations as well as with the group of orders that should be ready to process at each particular station at any moment in time. This should be done by deciding on the sequence in which they will be

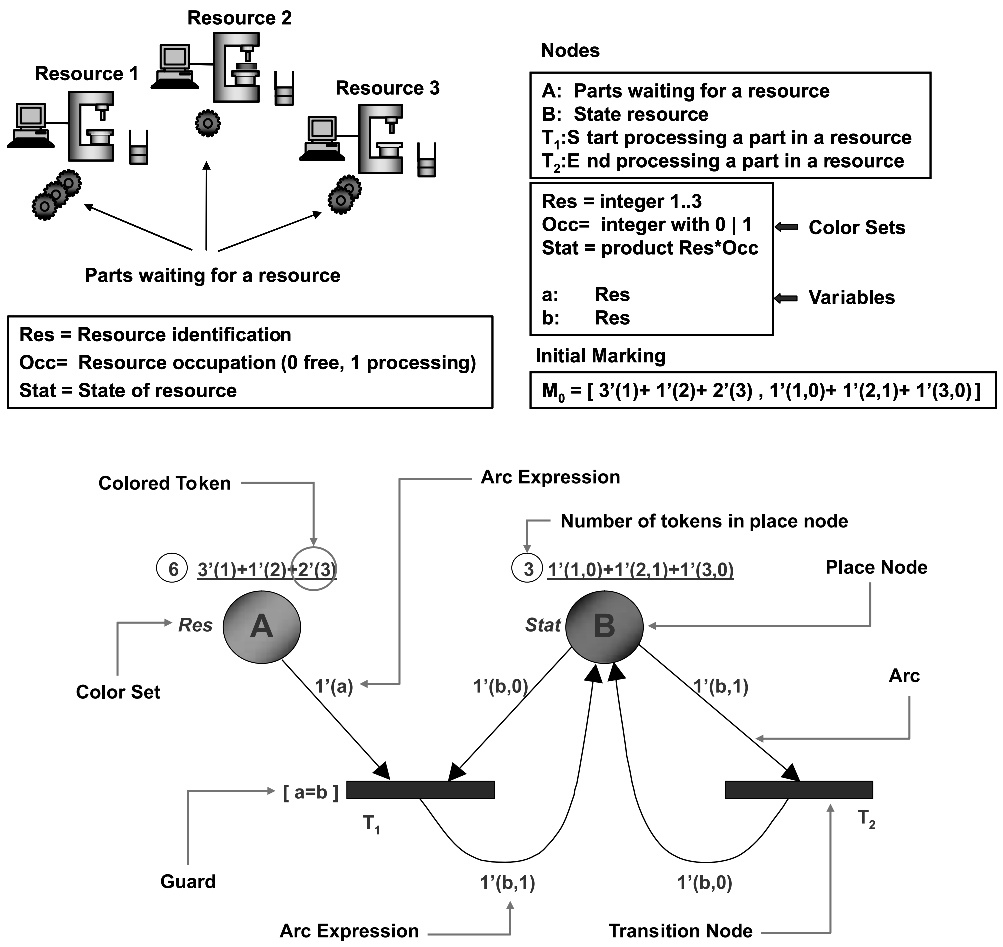


Figure 1. Graphical components of a CPN model

executed and calculating the initial and final times resulting from each operation.

Knowledge about the behavior of a DES is not easily acquired by simply analyzing output data, but by also understanding the events and how they are related. Thus, a modeling formalism that represents these types of systems should support data/information and processes/activities relations specification. It should do so in such a way that the effects of an event can be easily predicted, in order to enable or disable the occurrence of other events (decisions/options). On the other hand, the stochastic, dynamic and asynchronous nature of logistics systems requires a modeling formalism that takes all of these characteristics into account, allowing for a representation of their behavior in line with possible configurations. It also must support the improvement and automatic extension of the model over time, as well as its maintenance, in order to reflect changes in the domain modeled.

A requirement for the development of the methodology is therefore to build a model that provides all of the events of the DES and the relations between them. This will serve to represent the behavior of the complete system, and to choose the best alternative at each step of the decision, as a means of reaching objectives through the best combination of time and using the system resources. What is considered most suitable for modeling or representing these features in this kind of methodology is the use of a knowledge representation technique. This technique should allow not only the incorporation of knowledge about the behavior of the system, but also about an analysis of all possible future events, while controlling the generation of the different system states.

In this regard, the CPN offers tools for knowledge representation that are needed to formalize either the attributes or characteristics of the entities that flow into the system, and also the properties that these entities must possess so that a certain event can take place. It has been demonstrated that they are good tools for modeling logistics systems owing to several qualitative advantages, such as the ability to contain both the static and dynamic struc-

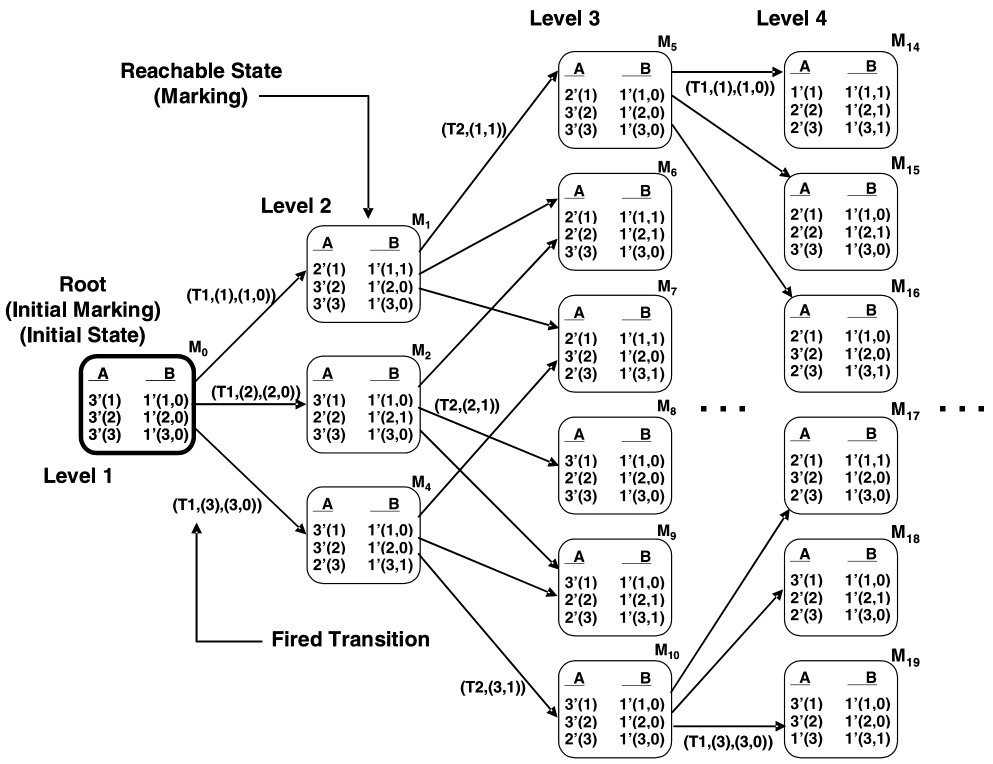


Figure 2. Reachability tree of a CPN (state space): analysis requirement

ture of the system, together with the physical layout constraints and its graphical nature. In addition, the CPN formalism is appropriate for modeling and visualizing behavioral patterns that show concurrency, synchronization and shared resources: key factors in decision-making in logistics systems.

The use of the CPN to formalize the behavior of DES can be regarded as a technique of knowledge representation since it allows the representation of entities and their state (places and colored tokens), events (transitions), behavior (*firing o*f the transitions) and meta-knowledge (guards). It also possesses the minimum set of conditions needed by any technique to represent knowledge: transparency, naturalness, clarity, efficiency, adequacy and modularity.

3.2 Analysis Tools Requirement

Once a model has been developed, its behavior must be analyzed in order to evaluate the different decision alternatives that can be put forward for solving a DES problem at any of the system's states. Simulation techniques can be used to carry out the analysis task. From the system model (knowledge representation), these techniques should allow the evaluation of their performance and the analysis of the different sequences of events that may occur (operation sequences, transportation sequences, etc.) The objective is to determine the most appropriate sequence of events or, likewise, to find an efficient solution to the decision problem.

Although current environments of simulation allow the automatic evaluation of different operation alternatives, in systems with a considerable number of scenarios that require evaluation, formalizing the model so that different scenarios can be specified through changes in parameter values is not possible. It is practically impossible to simulate all of the operation alternatives in order to provide a correct answer within a given time interval; analytical tools must also be used for evaluating all possible operation alternatives in order to select the best configuration of decision variables.

The proposed methodology uses an analysis method offered by the CPN formalism. The purpose of this method is to build the state space, also called an occurrence graph or reachability graph/tree, of a system modeled by a CPN. The basic idea behind a reachability tree is to build a graph that contains a node for each state (marking of a CPN) reachable in the modeled system, and an arc for each event occurrence (transition firing of a CPN), allowing a representation of the system's dynamic properties as shown in Figure 2.

Assume that we have a CPN model with two place nodes, A and B (Figure 1), two transition nodes, T_1

and T_2, and an initial marking $M_0 = [3'(1) + 3'(2) + 3'(3), 1'(1,0)+1'(2,0)+1'(3,0)]$ (i.e. three resources are free and there are three parts waiting for each resource):

- Each node in the tree represents a marking (state of the system: $M_0, M_1 \ldots, M_{ns}$; ns is the total number of states in the reachability tree), which is described by specifying the tokens in each place node of the CPN: the first column represents the tokens in place node A and the second column represents the tokens in place node B.
- Each arc represents the firing of a certain transition. The identification of the fired transition, together with the tokens used to enable the transition, is described by the text associated with the arc. The left component of the pair indicates the transition while the right component corresponds to token information.
- Graphically, the node with the thicker edge represents the initial marking.

The reachability tree determines the set of states $M = \{M_1, M_2, \ldots, M_{ns}\}$ which can be reached from the initial state M_0. The rules to generate the reachability tree are:

- The root of the tree is the initial state M_0.
- For each node of the tree, a *child node* is generated for each transition enabled for that state. The child nodes represent the state of the CPN once each of the enabled transitions is fired.
- When a child node is generated with a marking that already exists in the tree, this node is regarded as an *old node* and no new nodes are generated from it.
- A node that has no enabled transitions is regarded as a *leaf node*.
- A node that is not considered an old node, or leaf node, is considered a *new node* to indicate that it is possible to generate new child nodes.

This method of analysis, the formal definition of which is described extensively in [12], is the basis for state space exploration of DESs. Exploring the state space by generating the reachability tree of a CPN is part of the methodology that will provide solutions to optimization problems, which is described in Section 5.

3.3 Optimization Requirement

The efficient assignation of the values of the decision variables (optimization) which have been specified in the DES model requires that the system be described as a decision problem of OR. This problem must be solved to optimize the system operation. However, in generating the state space of real logistical systems, the number of states may grow to a size computationally prohibited. On the other hand, the computing time required to explore in an exhaustive manner a state space with these features restricts their use to academic systems. To control the exponential growth in the evaluation of possible alternatives, it is necessary to use effective search algorithms that allow the selection of a subset (best alternative) of possible system configurations. The simulation/optimization approach proposed in [30] justifies the use of heuristics as an obvious solution that can guide a search trying to improve the assignation of decision variables regarding certain performance measures of interest.

The basic idea underlying the dynamic generation of the state space of a DES, specified by the formalism of CPN, is to translate a problem of planning, scheduling or routing into a search problem in the state space of the system. This should be done in a way in which, through the use of a cost function and heuristics, it would be possible to obtain the best possible path from some initial state of the system (M_0) to a desired final state (M_f). This means finding a combination of events that could be considered a good solution (in the best case, the optimal solution) of the decision problem (see Figure 3).

4. Methodological Proposal

Although OR provides techniques to optimize the right sequence of operation to be assigned to different production resources, in many cases it is necessary to adjust the formulation of the decision model so that it can be solved by existing methods (formulating simplified decision models). Therefore, it is not possible to guarantee that the optimal solution of the model corresponds to the optimal solution of the problem. In this sense, the simulation has proved to be useful for evaluating the performance of different configurations and/or alternative operating procedures for complex logistics systems and production systems. However, when simulation techniques are applied, several limitations arise due to the inability to evaluate more than a fraction of the immense range of possible scenarios or options available.

Given such circumstances, it is possible to use techniques provided by AI for problem solving, for example, by searching through the state space. Nevertheless, the large number of decision variables in the current logistics systems often leads to a state space of considerable size, which makes its computational manipulation practically impossible. The information on the particular domain of the problem can often be used to assist in the search in the state space, in an efficient manner, by the use of heuristics. This requires the use of a formalism of knowledge representation, useful for decision-making in logistics [31, 32].

Figure 4 illustrates the methodological proposal for solving optimization problems by integrating features from the OR, AI and simulation research areas, using CPN

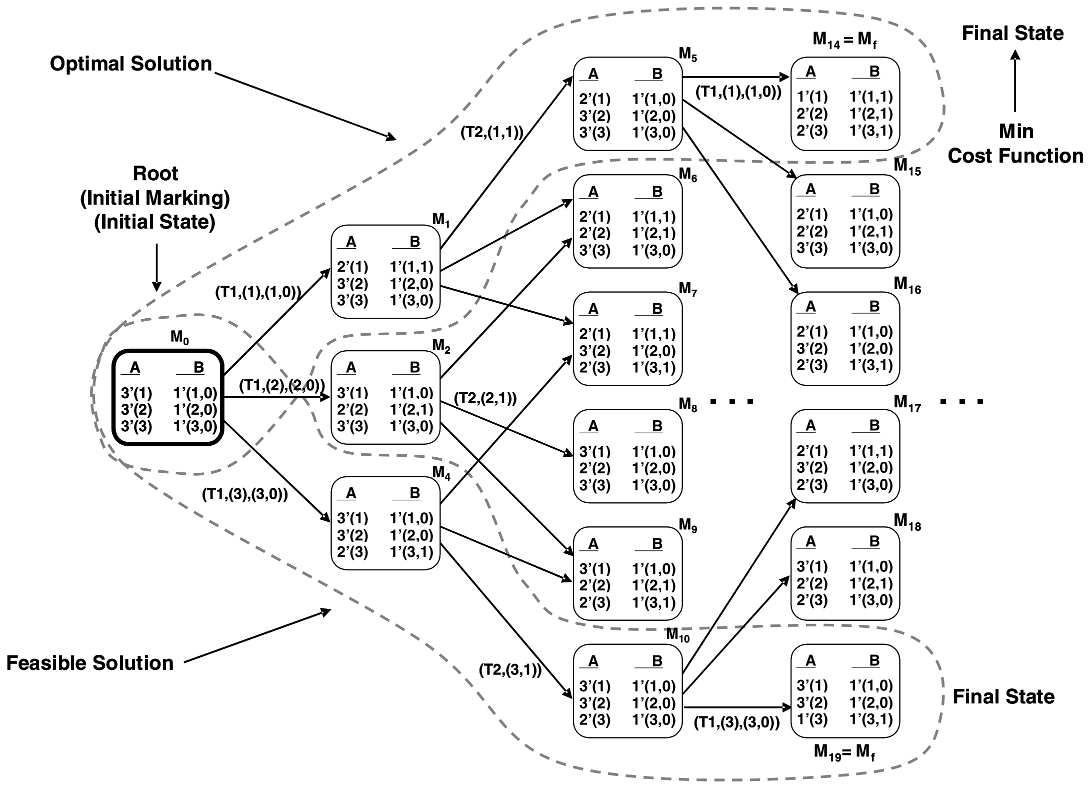

Optimal Solution
Root (Initial Marking) (Initial State)
Feasible Solution
Final State
Min Cost Function
Final State
M0
M1
M2
M4
M5
M6
M7
M8
M9
M10
M14 = Mf
M15
M16
M17
M18
M19= Mf
(T1,(1),(1,0))
(T1,(2),(2,0))
(T1,(3),(3,0))
(T2,(1,1))
(T2,(2,1))
(T2,(3,1))
(T1,(1),(1,0))
(T1,(3),(3,0))
A B 3'(1) 1'(1,0) 3'(2) 1'(2,0) 3'(3) 1'(3,0)
A B 2'(1) 1'(1,1) 3'(2) 1'(2,0) 3'(3) 1'(3,0)
A B 3'(1) 1'(1,0) 2'(2) 1'(2,1) 3'(3) 1'(3,0)
A B 3'(1) 1'(1,0) 3'(2) 1'(2,0) 2'(3) 1'(3,1)
A B 2'(1) 1'(1,0) 3'(2) 1'(2,0) 3'(3) 1'(3,0)
A B 2'(1) 1'(1,1) 2'(2) 1'(2,1) 3'(3) 1'(3,0)
A B 2'(1) 1'(1,1) 3'(2) 1'(2,0) 2'(3) 1'(3,1)
A B 3'(1) 1'(1,0) 2'(2) 1'(2,0) 3'(3) 1'(3,0)
A B 3'(1) 1'(1,0) 2'(2) 1'(2,1) 2'(3) 1'(3,1)
A B 3'(1) 1'(1,0) 3'(2) 1'(2,0) 2'(3) 1'(3,0)
A B 1'(1) 1'(1,1) 2'(2) 1'(2,1) 2'(3) 1'(3,1)
A B 2'(1) 1'(1,0) 2'(2) 1'(2,1) 3'(3) 1'(3,0)
A B 2'(1) 1'(1,0) 3'(2) 1'(2,0) 2'(3) 1'(3,1)
A B 2'(1) 1'(1,1) 3'(2) 1'(2,0) 2'(3) 1'(3,0)
A B 3'(1) 1'(1,0) 2'(2) 1'(2,1) 2'(3) 1'(3,0)
A B 3'(1) 1'(1,0) 3'(2) 1'(2,0) 1'(3) 1'(3,1)
. . .
. . .

Figure 3. Reachability tree of a CPN (state space): optimization requirement

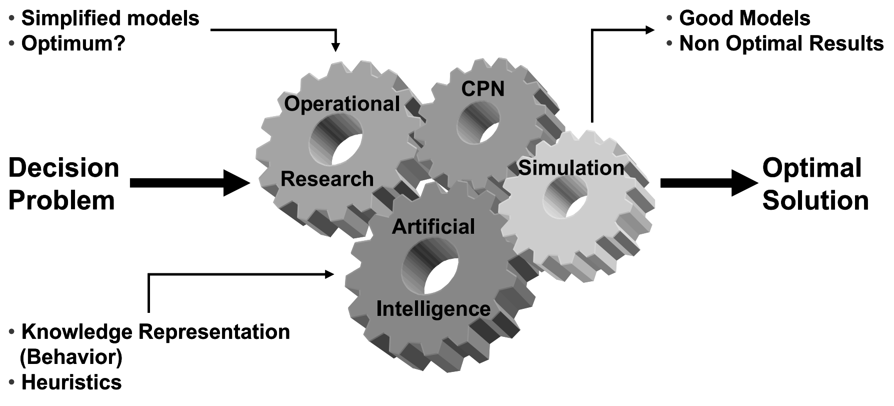


Figure 4. Integration of OR, AI and simulation, for the solution of optimization problems

as the knowledge representation formalism to model a decision problem. The CPN representation is used as the simulation model that allows a quantitative analysis of those scenarios that can lead to optimal solutions, while the evaluation of certain states is discarded (i.e. only the part of the state space that is worth exploring is analyzed):

- From the AI standpoint, concepts on structured formalisms of knowledge representation are used. These concepts allow inferences about the domain of the optimization problem, as well as concepts on solving problems through searches.

- From the simulation standpoint, it should support the evaluation of different possible scenarios of the decision problem. It is based on the evaluation of a model representing the knowledge about the domain of the optimization problem. For a CPN model, a branch of the tree will correspond to the simulation of a certain scenario of the modeling system.

- From the OR standpoint, the model should allow the efficient assignation of values to the decision variables (optimization). A cost function is evaluated in each state to guide the exploration of the optimization problem.

5. Description of the Methodology

The methodology is divided into six stages or phases, each of which is a step towards solving an optimization problem. The methodology is applicable to any combinatorial optimization problem or decision problem whose domain can be formulated as a DES, and is especially useful for solving logistical problems (transportation, scheduling, routing) and production planning problems.

Figure 5 illustrates the different phases of the methodology. The rectangles in the left column indicate the activity that should be carried out at each stage; the rectangles on the right indicate the result of the activity in each phase and provide input information to carry out the activities of the next phase. Next, a description of each stage of the methodology is provided, some of which are divided into several steps.

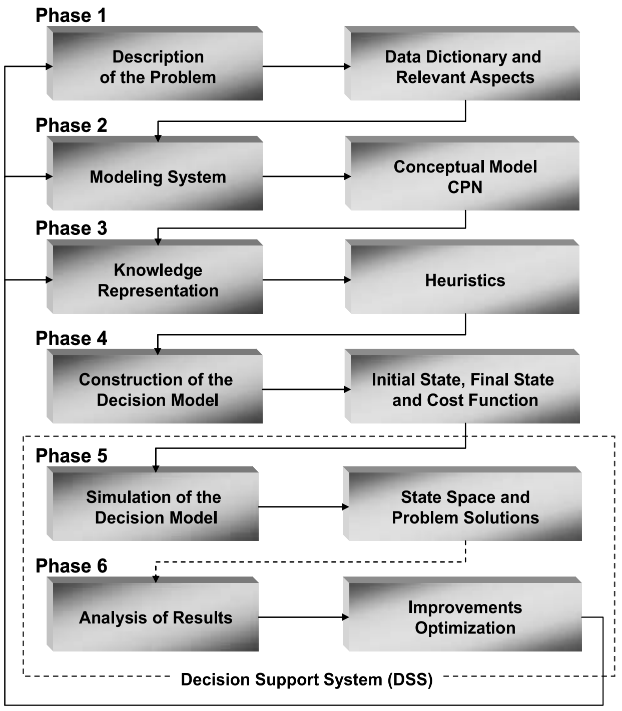

Phase 1
Description of the Problem
Data Dictionary and Relevant Aspects
Phase 2
Modeling System
Conceptual Model CPN
Phase 3
Knowledge Representation
Heuristics
Phase 4
Construction of the Decision Model
Initial State, Final State and Cost Function
Phase 5
Simulation of the Decision Model
State Space and Problem Solutions
Phase 6
Analysis of Results
Improvements Optimization
Decision Support System (DSS)

Figure 5. Phases of the methodology designed for solving optimization problems

5.1 Phase 1: Description of the Problem

To carry out this phase, it is necessary to understand the domain of the problem in order to abstract only the most relevant aspects for its resolution. This phase consists of three steps:

1. *Abstraction of the problem as a DES*: During this step of the methodology, the decision problem should be described as a system whose properties of interest only change in a sequence of time instants (occurrence of events), whereas they remain constant the rest of the time. The data and relevant aspects of the problem must then be classified into one of the following groups: temporary entities, resources, attributes, activities, duration of the activities, events, queues and initial and final states of the system.

2. *Determination of the objectives of the problem*: The objectives to be achieved by solving the optimization problem must be observable, precise, reasonable, understandable, measurable, and should be expressed in terms of some of the final states of the system described in the previous step.

3. *Determination of the dynamics of interest of the DES*: The dynamics of a DES are determined by the possible events indicating the beginning or end of an activity that will represent different changes of state. The changes in the states of the system take place at the instant of time that an event occurs. Determining the dynamics of interest can be quite difficult as it may depend on many factors. Although it should always be represented in the most simplified way possible, it should accurately reflect the criteria for obtaining a solution to the problem that has to be solved. During this step, the following tasks should be performed:

   - Create a list of activities (with its duration time) and events of interest that can generate a change in the state of the system. A state of the system is specified by describing the state of resources, temporary entities, queues and storage units.
   - Create a list of all of the resources or permanent entities of the system, temporary entities, queues and storage units.
   - Create a list of the conditions that must be satisfied in order to ensure that specified events may occur, i.e. the conditions an activity needs to be performed. These conditions are called preconditions.
   - Create a list specifying the effects on the state variables (state change) that must take place

after the occurrence of different events. These effects are called postconditions.

- Determine those aspects on which decisions can be made, differentiating between those aspects that are considered sensitive to the decisions.

At the end of this phase, there will be a specification of all of the events, resources, temporary entities, queues and storage units of the DES. It will also describe the preconditions and postconditions of each of the events and aspects for the decision-making considered relevant to solve the problem.

5.2 Phase 2: Modeling System

At this phase, the description of the decision problem obtained in the earlier phase is transformed into a conceptual model that describes the behavior or dynamics of the DES. The conceptual model must specify the most important structural relations of the system under study, taking into account the objective of the problem. It also should collect, in a detailed and precise manner, the dynamic relationships among the different elements.

In this phase events, resources, temporary entities, queues and storage units of the DES are represented by the CPN formalism. This modeling phase consists of three steps:

1. *DES relations specification*: This step defines the major dynamic relations that drive the system behavior together with the cause–effect relationships that describe how the system is affected by the decision-making activity.

2. *Graphical formalization of the events relationship*: This step formulates the CPN model that will describe the behavior of the DES. Listed below are some guidelines for its construction:

   - The queues, resources and temporary entities can be represented as places of the CPN.
   - The state of the resources and temporary entities can be represented by colored tokens in the places.
   - The state variables can be associated to the attributes of the entities represented as colors of the tokens.
   - The events and/or activities can be represented as transitions.
   - The preconditions can be represented by the arcs at the entrance to the transitions, and their corresponding arc expressions.
   - The postconditions can be represented by the output arcs of the transitions and their corresponding arc expressions.

3. *Analysis of the effects of possible decisions*: With the CPN model obtained in the previous step, this step will analyze those properties of the system that are not directly related to its evolution in the temporary domain, but which determine their dynamics, such as the existence of deadlocks in the system, or the set of possible states that the system can reach from a certain operational context. Through the analysis of the CPN, it is possible to detect unwanted states that can be avoided through the use of heuristics and metaheuristics (see phase 3). In addition, during this step it is possible to modify the CPN in order to avoid interlocking in the dynamics of a system.

At the end of this phase there will be a conceptual model, formalized through a CPN, that represents the structure and dynamics or behavior of the DES.

5.3 Phase 3: Knowledge Representation

The analysis of a CPN through the generation of the reachability tree allows heuristics and metaheuristics to be designed to reduce the state space, avoiding the firing of certain transitions (events) under certain conditions. Hence, at this phase the expert knowledge should be specified in order to design heuristics and metaheuristics that allow for the conditioning of the occurrence of certain events to avoid unwanted states. These heuristics and metaheuristics will be incorporated into the CPN model through the use of guards.

At the end of this phase, a set of conditions in the firing of one or more events will be obtained and properly represented by guards in the transitions that represent these events.

5.4 Phase 4: Construction of the Decision Model

In the context of OR, each decision problem presents certain basic components: decision variables, objective function, constraints, functional relationships and parameters. A *decision problem* model simultaneously expresses the objective function, the decision variables and constraints in a single mathematical model, along with the functional relationships involved and their parameters. On the other hand, the optimization is the process of trying to find the best possible solution for a particular problem. In an optimization problem there may exist different solutions. Therefore, there must be a criterion to discriminate among them, with the purpose of selecting the best one.

In the context of AI, the first step for solving a problem is the formulation of an *objective* based on a given situation. It is considered that an objective is a set of real-world states that satisfies different requirements, enough to be considered acceptable solutions to the problem. The resolution of the problem is to find what sequence of actions allows the objective of a state to be achieved, also called the final state. Thus, one problem can be defined by four components. The first is the *initial state* (initial situation of relevant aspects of the problem). The second is related to a description of the possible *actions* available (transformation functions or set of transition operators that can be applied to a state to get to the next state or successor state). The third is the *objective test* (it determines whether a state is a state objective or a final state). The last is related to the *cost function* of the path (it assigns a numerical cost to each path, which reflects a performance measure).

Implicitly, the initial state and the operators (or successor function) define the state space of the problem. The state space allows the formal description of a problem as a set of transformations. These transformations go from a given situation to some desired situations, through a series of allowed operations. The state space creates a graph in which nodes are states and the arcs between the nodes are actions. A *path* in the state space is a sequence of states connected by a sequence of actions (a collection of states of the problem). One *solution* to a problem is the route from the initial state to an objective state, and the process of finding this sequence is called the *search*. A search algorithm takes a problem as an input and returns a solution in a manner of sequence of actions. The quality of the solution is measured by the cost function of the path. An *optimal solution* has the lowest cost for the path among all of the solutions [8, 32].

At this phase of the methodology, the conceptual model obtained in phase 2 and the conditions obtained in phase 3 are transformed into a decision problem model that may be solved as a search problem, which requires the completion of the following steps:

1. *Specifying the initial state*: The initial state represents the original situation or the initial conditions of a problem. In the case of a DES, it represents the initial state of the resources, temporary entities, queues, storage units, etc. In the corresponding CPN model, the initial state corresponds to the initial configuration of tokens (number of tokens and their color) that is given by the expressions of initialization in each of the places that compose the model.

2. *Specifying the optimization objective*: The optimization objective (objective test) will make it possible to determine whether a state of the problem is a state objective or a final state. In a DES, the optimization objective represents one or more states that are desired for the resources, temporary entities, queues, storage units, etc, which are considered to be solutions to the decision problem. In the corresponding CPN model, the final state corresponds to one or more markings, which, depending on configuration and color of the tokens in each of the different places that compose the model, are considered solutions to the decision problem. It is possible to use wild cards (*) in the specification of a final state, when the color of one or more attributes of the marking is not relevant to the objective optimization. This is the case of those state variables that have not been considered as decision variables.

3. *Specifying a cost function*: A cost function will make it possible to assign a numerical cost to each state of the DES, during the generation of state space that reflects a performance measure (time, resource utilization, status of queues, etc.). To specify the cost function, it is necessary to determine which of the state variables specified in step 2 of phase 2 will be considered as decision variables, in order to solve the optimization problem as a search problem. Such a choice will depend on the performance measure that is to be optimized.
   In order to assign values to the decision variables and minimize the cost function, it is necessary to specify in the model the penalties for reaching certain states. In a CPN, this information can be formulated through a cost function that weighs the value of the decision variables (value of certain color attributes of the tokens) in the different places. This should be done in such a way that the function assigns high costs to those markings that correspond to unwanted (high times, unused resources, long queues, etc.) states (values of the decision variables).

Once the three steps described above have been completed, the decision model of the problem will be obtained, with the following properties:

- The decision variables are certain state variables of the conceptual model, selected according to the performance measure that needs to be optimized. In this methodology, the decision variables correspond to certain attributes of the tokens in the different places of the CPN model.

- The objective function is the cost function to be minimized.

- The restrictions or limitations on the values that can be assigned to the decision variables are specified by the different values of the attributes (color) that the tokens, in different final states already specified, can take.

- The functional relationships are the guards and the input and output expressions of the transitions.

- This model can be solved as a search problem since it presents all of the components:

- The initial state is the initial marking represented by the expressions of initialization of each place in the CPN model.
- Possible actions available (transformation functions or set of transition operators) are the transitions of the CPN.
- The minimum requirements or preconditions established in the formulation of the problem are the arc expressions at the entrance to the transitions, whose evaluation determines the transitions that are enabled at any given time for a determined state (marking).
- The objective test is the optimization objective, represented by the final state (or final states) specified for the DES.
- The cost function of the path (evaluation function) is the cost function specified to weigh the value of certain attributes of the color of the tokens on different places.

The solution of this model will consist of finding the values of the decision variables that simultaneously satisfy all of the restrictions (feasible solution for the model) and reach the objective that has been set. This means giving the minimum value to the cost function with respect to all other feasible solutions (optimal solution). Once the model is solved as a search problem, a solution will then be a path from the initial state to a final state, and an optimal solution will be the one that has the lowest cost among all solutions.

The cost function, used to determine the optimality of the solutions found in the problem solving, can also be used as a heuristics function to help efficiently search the state space [32].

At the end of this phase, a decision model that can be solved through a search problem will be generated.

5.5 Phase 5: Simulation of the Decision Model

In this phase, an automatic evaluation of the different scenarios or possible configurations of the modeling system should be performed by simulating the decision model obtained in the previous phase. This requires the generation of the state space of the problem, which is achieved by constructing the CPN reachability tree.

In order to support this phase of the methodology, a Decision Support System (DSS) has been developed [33]. This tool automatically generates the reachability tree of a CPN and also makes it possible to find feasible solutions for the decision model. This takes place by the use of the depth-first search strategy, and in many cases it leads to the finding of optimal solutions by using the cost function as an evaluation function for the best first search strategy.

At the end of this phase, the feasible and/or optimal solutions to the optimization problem would be identified. These solutions will be represented as a path from the initial state to the final state in terms of sequences of transitions (events) that enable the reaching of a desired final state from a certain initial state.

5.5.1 Decision Support System

The computer tool developed, a DSS to determine optimal plans, evaluates diverse heuristic search strategies in order to find the best configuration of the decision variables of a logistic problem, which could be conceived as a DES and modeled using the CPN formalism.

The CPN formalism is used to represent the system, and a DES, to dynamically build the state space, has been developed as a technique to automatically evaluate the system behavior. Based on the CPN formal definitions, an object-oriented application has been designed, in which different classes have been implemented to support CPN entities, Thus, a performance model can be represented by the following:

- The CPN model is described in an ASCII file using a specific language, specially designed for this tool, in which time activities can be described by a deterministic function.
- Initial state, which describes the initial conditions of the system that are specified by the initial marking in the CPN ASCII file.
- The optimization goal is specified by one or more desired end states using markings in the CPN. It supports the use of the wildcard ('*') to specify non-relevant marking information.
- The cost function is specified by certain values attached to different colors in each node place.

The DSS, developed in Visual C++ 6.0, interprets the CPN (i.e. ASCII file), the initial and final states, and the cost function described in the specification file of the model (CPN encoded), as well as generates and analyses the reachability tree (state space). As a result, the simulator will provide the sequence obtained to reach a final state from the initial state. Figure 6 illustrates a DSS scheme.

5.6 Phase 6: Analysis of Results

As a result of the previous phase, it is possible to obtain the set of feasible solutions and, consequently, the global optimum of the optimization problem. However, in some cases, a large number of state variables (or a high number of combinations of values of the state variables) can convert an optimization problem into a combinatorial optimization problem that is NP-hard. The solution of this problem may be impossible to determine in a reasonable

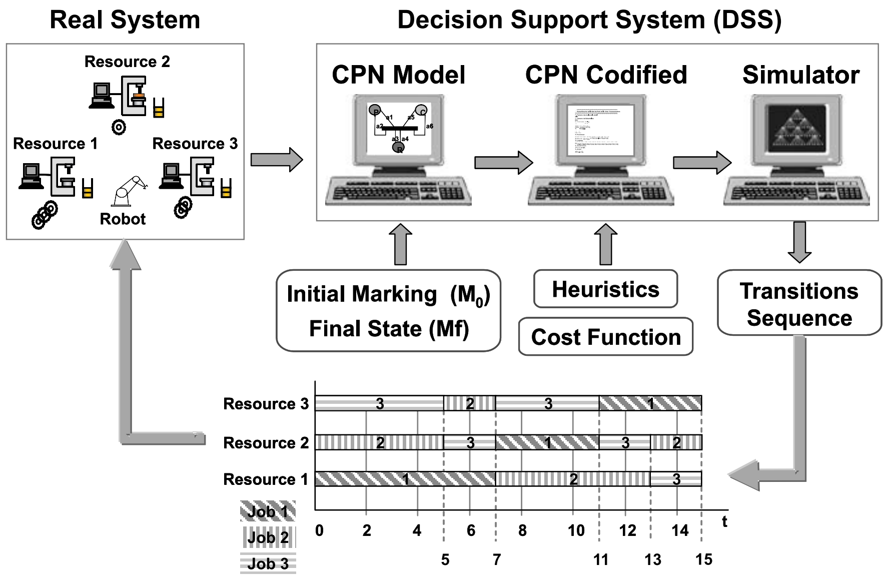


Figure 6. DSS scheme for solving optimization problems

computational time due to the exponential growth in the size of the state space.

This phase of the methodology consists of analyzing the state space generated in the previous phase, for the purpose of discovering unwanted states that could be avoided by improving the heuristics proposed in phase 3. By modifying the existing heuristics and/or by formulating new heuristics, it is possible to reduce the state space. This could be done in such a way that, through the simulation of the model, it will be possible to find an optimal solution to the problem.

Figure 7 represents the analysis process which consists of checking the generated state space. The goal is to modify/change the CPN model to reduce the size of the state space of a complex system by relaxing or introducing new constraints. The ability to properly update constraints in the CPN model by means of guard expressions is a key aspect to reduce the state space. Introducing/modifying new constraints is straightforward using the graphical representation of CPN.

6. Case Study

This section presents a case study that illustrates the results by implementing the different phases of the methodology described. The proposed system is a well-accepted benchmarking of the Job Shop Scheduling Problem (JSSP) [34–41].

6.1 Phase 1: Description of the Problem

The problem of scheduling tasks in JSSP includes a typology of NP-hard problems. This typology of problems

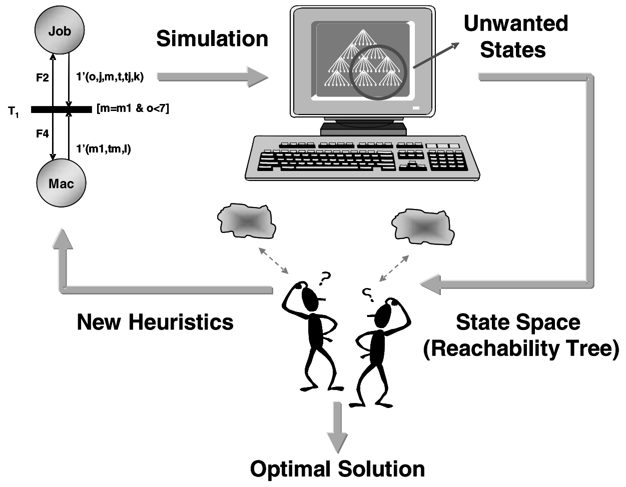


Figure 7. Scheme of the process of analysis of the simulation results

is considered one of the more difficult problems of combinatorial optimization to date. It consists of scheduling a group of jobs in a set of machines, subject to the restriction that each machine can process only one job at a time and each job has a specific processing order in each machine. The goal is to schedule the job to minimize the completion time of all jobs (makespan). The example shown in [30] is an instance of such problems (6 jobs and 6 machines, 6 × 6). However, there is a high number of instances for different numbers of jobs and machines (OR-Library [42]), considered as indispensable benchmark-

Table 1. Job sequence operations

Job	Operation 1		Operation 2		Operation 3		Operation 4		Operation 5	
	Mac	Time	Mac	Time	Mac	Time	Mac	Time	Mac	Time
1	1	72	0	87	4	95	2	66	3	60
2	4	5	3	35	0	48	2	39	1	54
3	1	46	3	20	2	21	0	97	4	55
4	0	59	3	19	4	46	1	34	2	37
5	4	23	2	73	3	25	1	24	0	28
6	3	28	0	45	4	5	1	78	2	83
7	0	53	3	71	1	37	4	29	2	12
8	4	12	2	87	3	33	1	55	0	38
9	2	49	3	83	1	40	0	48	4	7
10	2	65	3	17	0	90	4	27	1	23

Table 2. Data dictionary and relevant aspects of the problem

Data description	Type
Processing machines	Resources
Each job	Entity
Processing each task of a job in a machine	Activity
Start/end processing a task of a job in a machine	Dynamic of interest
The machine required to process each task of a job should be free (state of the entity machine)	Precondition
Information on the next task that a job must perform: order of sequence, machine and time of the next operation of the job (state of the entity job)	Postcondition
Information on the cumulative processing time spent by each machine	Postcondition
Information on the state of the entities: machines and jobs	System state
Minimum total time to process all jobs	Objective
Sequence for assigning tasks to machines	Decision

ings to test algorithms and combinatorial optimization methods.

The case study that has been selected is related to a problem of scheduling 10 jobs and 5 machines (la05 instance [42], 10 × 5). Table 1 shows the order and processing time for each job on each machine. The *Job* column indicates the number of the job (from 1 to 10), and each *Operation* heading (from 1 to 5) indicates both the machine (column *Mac*) and the average processing time of the job on the indicated machine (column *Time*).

The optimization objective is then to determine the sequence of assigning jobs to machines in such a way that the makespan is minimal, which for this problem is known as 593 units of time (under deterministic conditions), considering that the jobs may be processed in a deterministic time, neglecting the transportation times.

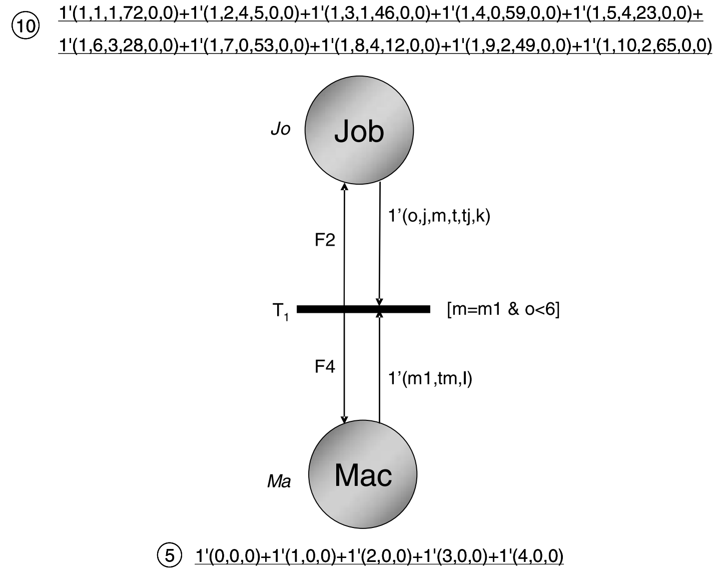
In this system, the activities would be the processing of each one of the tasks pertaining to a job in each machine (*Operation i*, *i* = 1, 2, 3, 4, 5), and the resources would be the machines. Table 2 describes the data and relevant aspects of the problem.

6.2 Phase 2: System Modeling

The application of phase 2 of the methodology to this scheduling problem is summarized in Tables 3, 4 and 5. They describe the colors, places, transitions and state variables of the CPN model that represent the production system Job Shop, which is shown in Figure 8. The whole system can be represented by a CPN model consisting of just two place nodes (machines and jobs) and one transition that represents the processing of a job by one machine. The preconditions to fire the transition are the availability of the machine (arc expression *1'(m1,tm,l)*) required by the job (*m* value of arc expression *1'(o,j,m,t,tj,k)*), and a processing order not exceeding 5, which is verified by the guard *[m=m1 & o<6]* (the symbol "&" corresponds to the "AND" logical operator).

In this example, and in order to simplify the representation of the dynamics of interest, the beginning and end of the activity has been modeled on the same transition. The duration of the activity is considered as an attribute of the entity *job* (see Tables 3, 4 and 5).

Information on the accumulated times (of the machines and jobs) have been added to the model through the variables *tj*, *k*, *tm* and *l*. These variables will be used in the



```
F2 = if j=1 & o=1 & l>k then 1'(2, 1, 0, 87, tj+t, l+t) else if j=1 & o=1 & l<=k then 1'(2, 1, 0, 87, tj+t, k+t)
else if j=1 & o=2 & l>k then 1'(3, 1, 4, 95, tj+t, l+t) else if j=1 & o=2 & l<=k then 1'(3, 1, 4, 95, tj+t, k+t)
else if j=1 & o=3 & l>k then 1'(4, 1, 2, 66, tj+t, l+t) else if j=1 & o=3 & l<=k then 1'(4, 1, 2, 66, tj+t, k+t)
else if j=1 & o=4 & l>k then 1'(5, 1, 3, 60, tj+t, l+t) else if j=1 & o=4 & l<=k then 1'(5, 1, 3, 60, tj+t, k+t)
else if j=2 & o=1 & l>k then 1'(2, 2, 3, 35, tj+t, l+t) else if j=2 & o=1 & l<=k then 1'(2, 2, 3, 35, tj+t, k+t)
else if j=2 & o=2 & l>k then 1'(3, 2, 0, 48, tj+t, l+t) else if j=2 & o=2 & l<=k then 1'(3, 2, 0, 48, tj+t, k+t)
else if j=2 & o=3 & l>k then 1'(4, 2, 2, 39, tj+t, l+t) else if j=2 & o=3 & l<=k then 1'(4, 2, 2, 39, tj+t, k+t)
else if j=2 & o=4 & l>k then 1'(5, 2, 1, 54, tj+t, l+t) else if j=2 & o=4 & l<=k then 1'(5, 2, 1, 54, tj+t, k+t)
else if j=3 & o=1 & l>k then 1'(2, 3, 3, 20, tj+t, l+t) else if j=3 & o=1 & l<=k then 1'(2, 3, 3, 20, tj+t, k+t)
else if j=3 & o=2 & l>k then 1'(3, 3, 2, 21, tj+t, l+t) else if j=3 & o=2 & l<=k then 1'(3, 3, 2, 21, tj+t, k+t)
else if j=3 & o=3 & l>k then 1'(4, 3, 0, 97, tj+t, l+t) else if j=3 & o=3 & l<=k then 1'(4, 3, 0, 97, tj+t, k+t)
else if j=3 & o=4 & l>k then 1'(5, 3, 4, 55, tj+t, l+t) else if j=3 & o=4 & l<=k then 1'(5, 3, 4, 55, tj+t, k+t)
else if j=4 & o=1 & l>k then 1'(2, 4, 3, 19, tj+t, l+t) else if j=4 & o=1 & l<=k then 1'(2, 4, 3, 19, tj+t, k+t)
else if j=4 & o=2 & l>k then 1'(3, 4, 4, 46, tj+t, l+t) else if j=4 & o=2 & l<=k then 1'(3, 4, 4, 46, tj+t, k+t)
else if j=4 & o=3 & l>k then 1'(4, 4, 1, 34, tj+t, l+t) else if j=4 & o=3 & l<=k then 1'(4, 4, 1, 34, tj+t, k+t)
else if j=4 & o=4 & l>k then 1'(5, 4, 2, 37, tj+t, l+t) else if j=4 & o=4 & l<=k then 1'(5, 4, 2, 37, tj+t, k+t)
else if j=5 & o=1 & l>k then 1'(2, 5, 2, 73, tj+t, l+t) else if j=5 & o=1 & l<=k then 1'(2, 5, 2, 73, tj+t, k+t)
else if j=5 & o=2 & l>k then 1'(3, 5, 3, 25, tj+t, l+t) else if j=5 & o=2 & l<=k then 1'(3, 5, 3, 25, tj+t, k+t)
else if j=5 & o=3 & l>k then 1'(4, 5, 1, 24, tj+t, l+t) else if j=5 & o=3 & l<=k then 1'(4, 5, 1, 24, tj+t, k+t)
else if j=5 & o=4 & l>k then 1'(5, 5, 0, 28, tj+t, l+t) else if j=5 & o=4 & l<=k then 1'(5, 5, 0, 28, tj+t, k+t)
else if j=6 & o=1 & l>k then 1'(2, 6, 0, 45, tj+t, l+t) else if j=6 & o=1 & l<=k then 1'(2, 6, 0, 45, tj+t, k+t)
else if j=6 & o=2 & l>k then 1'(3, 6, 4, 5, tj+t, l+t) else if j=6 & o=2 & l<=k then 1'(3, 6, 4, 5, tj+t, k+t)
else if j=6 & o=3 & l>k then 1'(4, 6, 1, 78, tj+t, l+t) else if j=6 & o=3 & l<=k then 1'(4, 6, 1, 78, tj+t, k+t)
else if j=6 & o=4 & l>k then 1'(5, 6, 2, 83, tj+t, l+t) else if j=6 & o=4 & l<=k then 1'(5, 6, 2, 83, tj+t, k+t)
else if j=7 & o=1 & l>k then 1'(2, 7, 3, 71, tj+t, l+t) else if j=7 & o=1 & l<=k then 1'(2, 7, 3, 71, tj+t, k+t)
else if j=7 & o=2 & l>k then 1'(3, 7, 1, 37, tj+t, l+t) else if j=7 & o=2 & l<=k then 1'(3, 7, 1, 37, tj+t, k+t)
else if j=7 & o=3 & l>k then 1'(4, 7, 4, 29, tj+t, l+t) else if j=7 & o=3 & l<=k then 1'(4, 7, 4, 29, tj+t, k+t)
else if j=7 & o=4 & l>k then 1'(5, 7, 2, 12, tj+t, l+t) else if j=7 & o=4 & l<=k then 1'(5, 7, 2, 12, tj+t, k+t)
else if j=8 & o=1 & l>k then 1'(2, 8, 2, 87, tj+t, l+t) else if j=8 & o=1 & l<=k then 1'(2, 8, 2, 87, tj+t, k+t)
else if j=8 & o=2 & l>k then 1'(3, 8, 3, 33, tj+t, l+t) else if j=8 & o=2 & l<=k then 1'(3, 8, 3, 33, tj+t, k+t)
else if j=8 & o=3 & l>k then 1'(4, 8, 1, 55, tj+t, l+t) else if j=8 & o=3 & l<=k then 1'(4, 8, 1, 55, tj+t, k+t)
else if j=8 & o=4 & l>k then 1'(5, 8, 0, 38, tj+t, l+t) else if j=8 & o=4 & l<=k then 1'(5, 8, 0, 38, tj+t, k+t)
else if j=9 & o=1 & l>k then 1'(2, 9, 3, 83, tj+t, l+t) else if j=9 & o=1 & l<=k then 1'(2, 9, 3, 83, tj+t, k+t)
else if j=9 & o=2 & l>k then 1'(3, 9, 1, 40, tj+t, l+t) else if j=9 & o=2 & l<=k then 1'(3, 9, 1, 40, tj+t, k+t)
else if j=9 & o=3 & l>k then 1'(4, 9, 0, 48, tj+t, l+t) else if j=9 & o=3 & l<=k then 1'(4, 9, 0, 48, tj+t, k+t)
else if j=9 & o=4 & l>k then 1'(5, 9, 4, 7, tj+t, l+t) else if j=9 & o=4 & l<=k then 1'(5, 9, 4, 7, tj+t, k+t)
else if j=10 & o=1 & l>k then 1'(2, 10, 3, 17, tj+t, l+t) else if j=10 & o=1 & l<=k then 1'(2, 10, 3, 17, tj+t, k+t)
else if j=10 & o=2 & l>k then 1'(3, 10, 0, 90, tj+t, l+t) else if j=10 & o=2 & l<=k then 1'(3, 10, 0, 90, tj+t, k+t)
else if j=10 & o=3 & l>k then 1'(4, 10, 4, 27, tj+t, l+t) else if j=10 & o=3 & l<=k then 1'(4, 10, 4, 27, tj+t, k+t)
else if j=10 & o=4 & l>k then 1'(5, 10, 1, 23, tj+t, l+t) else if j=10 & o=4 & l<=k then 1'(5, 10, 1, 23, tj+t, k+t);
else if o=5 & l>k then 1'(6, j, 0, 0, tj+t, l+t) else if o=5 & l<=k then 1'(6, j, 0, 0, tj+t, k+t);

F4= if l>=k then 1'(m1, tm+t, l+t) else if k>l then 1'(m1, tm+t, k+t);
```

Figure 8. The CPN model of the system Job Shop 10 × 5

Table 3. Color sets of the CPN model of the production system Job Shop 10 × 5

Color	Definition	Description	State variables
O	Integer 1..6	Sequence position of the next task of job *J* that must be processed	*o*
J	Integer 1..6	Job identifier	*j*
M	Integer 0..5	Machine in which the task number *O* of job *J* should be processed	*m*
T	Integer 1..10	Units of time required to process the task number *O* of job *J* in machine *M*	*t*
TJ	Integer 0..1000	Accumulated processing time of job *J*	*tj*
K	Integer 0..1000	Total time elapsed including waiting periods	*k*
Jo	Product *O*J*M*T*TJ*K*	Information describing the state of each job	
M1	Integer 0..5	Machine identifier	*m1*
TM	Integer 0..1000	Accumulated processing time of machine *M1*	*tm*
L	Integer 0..1000	Total elapsed time including waiting periods	*l*
Ma	Product *M1*TM*L*	Information describing the state of each machine	

Table 4. Places of the CPN model of the production system Job Shop 10 × 5

Place	Color	Description
Job	*Jo*	The tokens stored in the place *Job* represent the information that describes the state of each job in the production system. These are job identifier, machine where the next task should be executed, time required for each operation, order of the task in the sequence of operations required to complete the job, cumulative job processing time and total job processing time, including waiting periods.
Mac	*Ma*	The tokens stored in the place *Mac* represent the information that describes the state of each machine in the production system. These are machine identifier, cumulative machine processing time and total machine processing time, including waiting periods.

Table 5. Transitions of the CPN model of the production system Job Shop 10 × 5

Transition	Description
T_1	Processing of a task pertaining to a job in a machine.

Table 6. Total times without waiting times for the system jobs Job Shop 10 × 5

Job	1	2	3	4	5	6	7	8	9	10
Time	380	181	239	195	173	239	202	225	227	222

next phase to design a heuristic that will be incorporated into the CPN model by using guards.

The arc expression *F2* in Figure 8 represents the postcondition corresponding to the update of the information on the next operation that each job should perform (order, machine, and time for the next operation). The colors of the tokens will be updated according to the values shown in Table 1. If we consider, for example, that job *3* has finished the second operation ($m = 3, o = 2$), its colors will be updated to $o = 3$, $j = 3$, $m = 2$, $t = 21$. In a similar way, since the next processing operation should be performed in machine *0*, the new color values will be $o = 4$, $j = 3$, $m = 0$, $t = 97$.

The arc expression *F4* (see Figure 8) represents the postcondition corresponding to the update of the information on the machine where each operation is processed: processing time accumulated from the machine *m1* (variable *tm*) and total time including the cumulative time plus the waiting times (variable *l*).

It is worthwhile to note that when modifying only the initial marking, guard, and arc expression *F2*, the same generic CPN can be used to solve any instance of the JSSP.

6.3 Phase 3: Knowledge Representation

In this phase knowledge of the optimum makespan best recognized for this problem (593 units of time) will be considered. In general, it is quite common to make an initial estimate that requires improvement. Based on this knowledge, a heuristics is designed. This heuristics avoids states where the processing time for each job or the elapsed time for each machine exceeds 593 units of time.

Tables 6 and 7 show the total processing time for each job and the total processing time for each machine, respectively. These times are derived from Table 1 and are calculated by separately adding the times required for each job in each of the machines, and the times required by each machine to process each one of the jobs.

With this information and the information on the accumulated times that provide variables *tj*, *k*, *tm* and *l*, we can predict whether a job or machine might exceed the minimum makespan known. In this way, and by an empirical

Table 7. Total time without waiting times for the machine system Job Shop 10 × 5

Machine	0	1	2	3	4
Time	593	463	532	391	304

rule, the unwanted states can be avoided if it is known beforehand that a machine or a job exceed the minimum time known. This heuristic is incorporated to the CPN model modifying the guard:

$$[m = m1 \ \& \ o < 6] \tag{1}$$

in Figure 8 by incorporating different logical expressions. The expression

$$[(k + t) < 594] \tag{2}$$

indicates that the cumulative time of a job, including waiting times, may not exceed the value 593. Similarly, the expression

$$[(l + t) < 594] \tag{3}$$

indicates that the cumulative time of a machine, including waiting times, may not exceed the value 593. Thus, adding the expressions (2) and (3) to expression (1), the new guard would be

$$\begin{aligned} &[m = m1 \ \& \ o < 6] \ \& \ [(k + t) < 594] \\ &\& \ [(l + t) < 594] \end{aligned} \tag{4}$$

6.4 Phase 4: Construction of the Decision Model

The expressions of initialization of the CPN (underlined expressions associated with each place in Figure 8) indicate that in the initial state of the system the place *Job* has 10 tokens. Each one of them describes a job that is waiting to perform the first operation. The place *Mac* has five tokens, each one of them represents a machine that is initially free:

$$\begin{aligned} M_0 = \ & [1'(1, 1, 1, 72, 0, 0) + 1'(1, 2, 4, 5, 0, 0) \\ & + 1'(1, 3, 1, 46, 0, 0) + 1'(1, 4, 0, 59, 0, 0) \\ & + 1'(1, 5, 4, 23, 0, 0) + 1'(1, 6, 3, 28, 0, 0) \\ & + 1'(1, 7, 0, 53, 0, 0) + 1'(1, 8, 4, 12, 0, 0) \\ & + 1'(1, 9, 2, 49, 0, 0) + 1'(1, 10, 2, 65, 0, 0), \\ & 1'(0, 0, 0) + 1'(1, 0, 0) + 1'(2, 0, 0) \\ & + 1'(3, 0, 0) + 1'(4, 0, 0)] \end{aligned}$$

Marking M_f specifies the objective or final state to be achieved, that is, the state in which all jobs have finished processing their last task:

$$\begin{aligned} M_f = \ & [1'(6, 1, *, *, *, *) + 1'(6, 2, *, *, *, *) \\ & + 1'(6, 3, *, *, *, *) + 1'(6, 4, *, *, *, *) \\ & + 1'(6, 5, *, *, *, *) + 1'(6, 6, *, *, *, *) \\ & + 1'(6, 7, *, *, *, *) + 1'(6, 8, *, *, *, *) \\ & + 1'(6, 9, *, *, *, *) + 1'(6, 10, *, *, *, *), \\ & 1'(0, *, *) + 1'(1, *, *) + 1'(2, *, *) \\ & + 1'(3, *, *) + 1'(4, *, *)] \end{aligned}$$

The value of variable $o = 6$ in all tokens containing the marking M_f indicates that all jobs have completed their 5th operation. The wild card (*) in the marking indicates that the color of the corresponding attribute is indifferent.

A cost function has been formulated for this case study. This cost function weighs the order of tasks sequence that assigns a low cost to the first task of each job. It also assigns a cost a little higher to the second task of each job, and keeps increasing the cost as the order of tasks sequence for each job increases. Such a function ponders the order of tasks sequence, which is specified by the value assigned to the decision variable o in each token of the place *Job*, as represented in the following expression:

$$f(O_i) = \sum (W * O_i) \ \forall i = 1, 2, \cdots, 10. \tag{5}$$

The value of W is a numeric constant that indicates the desired weight of each task order, O. For example, given a certain state in which job 1 should process task 1, job 2 should process task 2, job 3 should process task 3, and so on, and assuming a cost of 2 units (i.e. $W = 2$), the cost of reaching the marking that represents this state is given by

$$\begin{aligned} f(O_i) &= (2 * 1) + (2 * 2) + (2 * 3) + (2 * 4) + (2 * 5) \\ &+ (2 * 6) + (2 * 7) + (2 * 8) + (2 * 9) + (2 * 10) \\ &= 2 + 4 + 6 + 8 + 10 + 12 + 14 + 16 + 18 + 20 \\ &= 110. \end{aligned} \tag{6}$$

This function assigns high costs to those states in which tasks of greater sequence order have been processed for some jobs, while tasks of lower sequence order have not yet been processed for other jobs. This function is employed as a heuristic function in order to avoid expanding the nodes of the search space in which some jobs have completed all of their tasks, while others remain unprocessed.

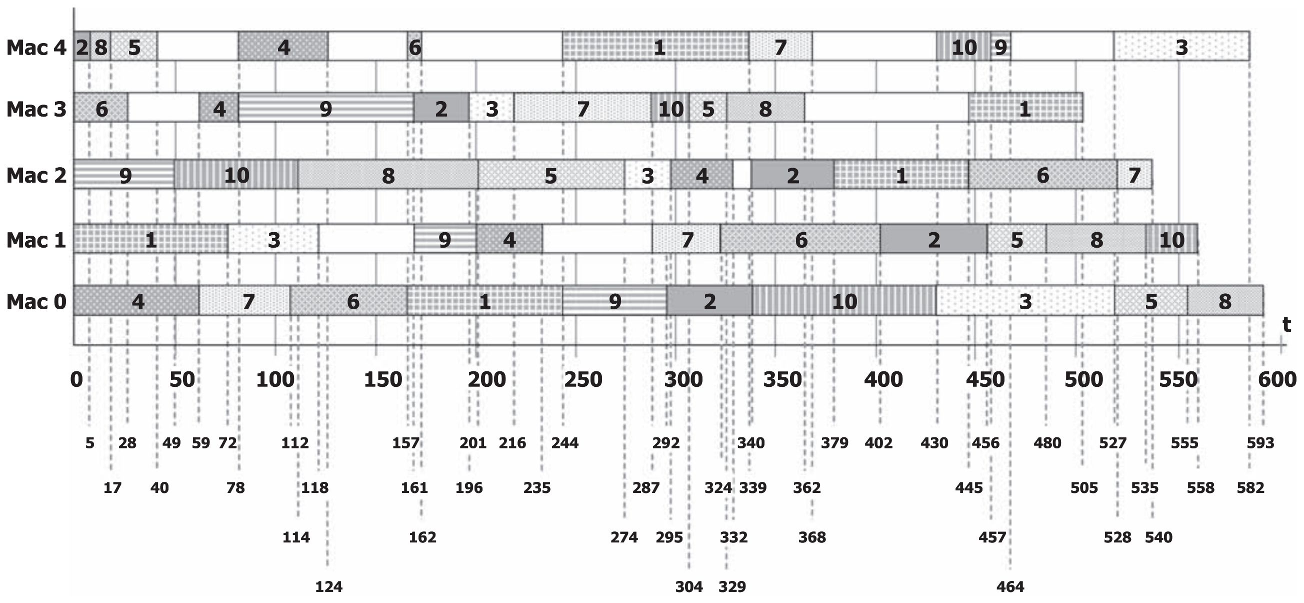


Figure 9. Gantt diagram representation of the optimal scheduling generated

6.5 Phase 5: Simulation of the Decision Model

As a result simulation in DSS, the scheduling can be obtained from the decision model specified in the previous phase. This scheduling is illustrated by a Gantt chart in Figure 9. The value of the makespan obtained is 593 units of time (optimal makespan). This case study has been using an adaptation of the heuristics Branch and Bound implemented in DSS and the cost function (6) specified in phase 4 as a heuristics function. The purpose is to reduce the search space.

The number of nodes explored to obtain this solution was 666.758 and the computation time was 2 hours 5 minutes and 41 seconds (on a PC with 2.00 GB RAM, 2.16 GHz Intel Core CPU on the MS Windows XP operating system). Although the solution found coincides with the optimal solution, the computational time is considered too high. In the next phase the CPN model is improved with a heuristic to reduce the computational time while generating the optimal solution.

6.6 Phase 6: Analysis of Results

By analyzing the state space generated in the previous phase, unwanted states that could be avoided have been detected. Thus, cumulative time, including waiting times for each job and each machine, should not exceed the values shown in Tables 6 and 7. By improving the heuristics proposed in phase 3 (expression (4)), it is possible to reduce the state space by neglecting the exploration of those unwanted states (i.e. nodes) that will not drive the system to the optimal goal state. Thus, it will be possible to find the optimal solution to the problem in a shorter computational time. The expression

$$(j = 1\,\&\,((380 - tj) + k) < 594 \\ \&\,((380 - tj) + l) < 594) \qquad (7)$$

indicates that the cumulative time, considering waiting times for the job 1 (variable k), plus the minimum time required to carry out operations that are needed to complete the processing of that job $(380 - tj)$, may not exceed the value 594. Similarly, the expression

$$(m = 0\,\&\,((593 - tm) + k) < 594 \\ \&\,((593 - tm) + l) < 594) \qquad (8)$$

indicates that the cumulative time including waiting times for machine 0 (variable l), plus the minimum time necessary for performing the operations that must processed in this machine $(593 - tm)$, may not exceed the value 594. This applies to all machines and all jobs as indicated in the expression

$$[(j = 1\,\&\,((380 - tj) + k) < 594\,\&\,((380 - tj) + l) \\ < 594)|(j = 2\,\&\,((181 - tj) + k) < 594\,\& \\ ((181 - tj) + l) < 594)|(j = 3\,\&\,((239 - tj) + k) \\ < 594\,\&\,((239 - tj) + l) < 594)|(j = 4\,\& \\ ((195 - tj) + k) < 594\,\&\,((195 - tj) + l) < 594)|$$

$$\begin{aligned}
&(j = 5 \,\&\, ((173 - tj) + k) < 594 \,\&\, ((173 - tj) + l) \\
&< 594) | (j = 6 \,\&\, ((239 - tj) + k) < 594 \,\& \\
&((239 - tj) + l) < 594) | (j = 7 \,\&\, ((202 - tj) + k) \\
&< 594 \,\&\, ((202 - tj) + l) < 594) | (j = 8 \,\& \\
&((225 - tj) + k) < 594 \,\&\, ((225 - tj) + l) \\
&< 594) | (j = 9 \,\&\, ((227 - tj) + k) < 594 \,\& \\
&((227 - tj) + l) < 594) | (j = 10 \,\& \\
&((222 - tj) + k) < 594 \,\&\, ((222 - tj) + l) < 594)] \,\& \\
&[(m = 0 \,\&\, ((593 - tm) + k) < 594 \,\&\, ((593 - tm) + l) \\
&< 594) | (m = 1 \,\&\, ((463 - tm) + k) < 594 \,\& \\
&((463 - tm) + l) < 594) | (m = 2 \,\&\, ((532 - tm) + k) \\
&< 594 \,\&\, ((532 - tm) + l) < 594) | (m = 3 \,\& \\
&((391 - tm) + k) < 594 \,\&\, ((391 - tm) + l) < 594) | \\
&(m = 4 \,\&\, ((304 - tm) + k) < 594 \,\&\, ((304 - tm) + l) \\
&< 594)] \qquad (9)
\end{aligned}$$

The guard of the transition represented in the CPN model (expression (4)) has been modified by adding these new logical expressions (9) to improve the search. The scheduling obtained once the simulation finalized was the same as that illustrated by a Gantt chart in Figure 9. The value of the makespan obtained was 593 units of time (optimal makespan). The number of nodes explored to obtain this solution was 2155 and the computation time was 8 seconds. Thus, the computational time required was shorter than the best time reported in this benchmarking problem using genetic algorithms [38].

7. Conclusions

In this paper we have presented a methodology to solve optimization problems of logistic systems in particular and DES in general. This methodology, along with a DSS, which supports the methodology, allows the automatic generation of the sequence of events and/or activities that should take place within a system, to reach a certain desired final state from a specified initial state, minimizing a certain cost function.

A very important benefit provided by the methodology is the ability to integrate different approaches and alternatives, developed by various areas of knowledge and always re-using the same problem model.

The methodology also facilitates the development of heuristics based on the system's behavior, incorporating expert knowledge for the generation of constraints that will decisively reduce the computational time needed to explore the state space. This feature is particularly relevant when working with a large number of decision variables or value combinations. It is thus possible to avoid exploring behavior alternatives that lead to unwanted states and consequently do not provide relevant information for the system's optimization.

The methodology has been used to solve various academic, logistics and production problems [20, 33, 43–45], among others. For these problems, optimal or quasi-optimal solutions have been found. They have reported computation times that range from a few seconds, where heuristics have been employed, to fewer than 12 hours, where the state space has been explored without the use of heuristics.

The developed software tool has been used mainly in the solution of academic and benchmarking problems by members of the research team. However, a new prototype based on a distributed platform is under development to support the analysis and optimization of industrial and logistics problems.

The methodology is applicable to any combinatorial optimization problem or decision problem whose domain can be formulated as a DES. Although CPN is not popular in industry, the continuous research efforts of the CPN community surely will contribute to extend its use as the basis for future decision support tools.

8. APPENDIX: Nomenclature

8.1 CPN Formalism

A = A finite set of arcs
A_i = An arc, $i = 1, \ldots, na$
c_r = The value of a color, $r = 1, \ldots, nr$
C = A color function
C_q = A color set, $q = 1, \ldots, nc$
E = An arc expression function
G = A guard function
i = Subscript for arcs
I = An initialization function
j = Subscript for place nodes
k = Subscript for transition nodes
M = Set of markings (states) which can be reached from the initial state M_0
M_f = Desired final state of the system
M_s = A marking, $s = 0, \ldots, ns$
M_0 = The initial marking (initial state of the system)
n = Number of identical tokens
na = The total number of arcs in CPN model
nc = The total number of color sets in CPN model
np = The total number of place nodes in CPN model
nr = Number of colored tokens attributes

ns = The total number of markings (states) in the reachability tree
nt = The total number of transitions nodes in CPN model.
N = A node function
P = A finite set of place nodes.
P_j = A place node, $j = 1, \ldots, np$
q = Subscript for color sets
r = Subscript for colored tokens attributes
s = Subscript for markings
T = A finite set of transition nodes
T_k = A transition node, $k = 1, \ldots, nt$
Σ = Finite set and not empty of color sets

8.2 Case Study

W = A numeric constant that indicates the desired weight of each task order O
$f(O_i)$ = A cost function that weighs the order of tasks sequence
i = Subscript for job identifiers
j = A variable of type (color) J
J = A color set: job identifiers
Jo = A color set: information describing the states of each job
Job = A place node: represent the information that describes the state of each job in the production system
k = A variable of type (color) K
K = A color set: total time elapsed including waiting periods
l = A variable of type (color) L
L = A color set: total elapsed time including waiting periods
m = A variable of type (color) M
M = A color set: machines in which the tasks of the jobs should be processed
Ma = A color set: information describing the states of each machine
Mac = A place node: represent the information that describes the state of each machine in the production system
$m1$ = A variable of type (color) $M1$
$M1$ = A color set: machine identifiers
o = A variable of type (color) O
O = A color set: sequence position of the next task of job J that must be processed
O_i = The decision variable o in each token of the place *Job*
t = A variable of type (color) T
T = A color set: units of time required to process the task number O of job J in machine M
tj = A variable of type (color) TJ
TJ = A color set: accumulated processing time of job J
tm = A variable of type (color) TM
TM = A color set: accumulated processing time of machine $M1$
T_1 = A transition node: represent the processing of a task pertaining to a job in a machine
$\&$ = The 'AND' logical operator
$|$ = The 'OR' logical operator

8. References

[1] Lee, S. and I.E. Grossmann. 2000. New algorithms for nonlinear generalized disjunctive programming. *Comput. Chem. Eng.*, **24**: 2125–2141.

[2] Grossmann, I.E., S.A. van den Heever and I. Harjunkoski. 2001. Discrete optimization methods and their role in the integration of planning and scheduling. *Comput. Chem. Eng.*, **28**: 1169–1192.

[3] Kallrath, J. 2002. Planning and scheduling in the process industry. *OR Spectrum*, **24**: 219–250.

[4] Biegler, L.T. and I.E. Grossmann. 2004. Retrospective on optimization. *Comput. Chem. Eng.*, **28**: 1169–1192.

[5] Méndez, C.A., J. Cerdá, I.E. Grossmann, et al. 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput. Chem. Eng.*, **30**: 913–946.

[6] Zanakis, S.H., J.R. Evans and A.A. Vazacopoulos. 1989. Heuristic methods and applications: a categorized survey. *Eur. J. Oper. Res.*, **43**: 88–110.

[7] Voß, S. 2001. Meta-heuristics: the state of the art. In *Local Search for Planning and Scheduling* (*Lecture Notes in Artificial Intelligence,* Vol. 2148), Springer, Berlin, pp. 1–23.

[8] Michalewicz, Z. and D.B. Fogel. 2002. *How to Solve It: Modern Heuristics*, Springer, Berlin.

[9] Silver, E.A. 2004. An overview of heuristic solution methods. *J. Oper. Res. Soc.,* **55**(9): 936–956.

[10] Kondili, E., C.C. Pantelides and R.W.H. Sargent. 1993. A general algorithm for short-term scheduling of batch operations-I. MILP formulation. *Comput. Chem. Eng.*, **17**(2): 211–227.

[11] Pantelides, C.C. 1994. Unified frameworks for the optimal process planning and scheduling. In *Proceedings on the Second Conference on Foundations of Computer Aided Operations*, pp. 253–274.

[12] Jensen, K. 1997. *Coloured Petri Nets: Basics Concepts, Analysis Methods and Practical Use*. Springer, Berlin.

[13] Jensen, K., L.M. Kristensen and L. Wells. 2007. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Trans.*, **9**(3): 213–254.

[14] Proth, J.M., and X. Xie. 1996. *Petri Nets: A Tool for Design and Management of Manufacturing Systems*. Wiley, New York.

[15] Tuncel, G. and G.M. Bayhan. 2007. Applications of Petri nets in production scheduling: a review. *Int. J. Adv. Manuf. Technol.*, **34**: 762–773.

[16] Chien, C.-F. and C.-H. Chen. 2007. Using genetic algorithms (GA) and a coloured timed Petri net (CTPN) for modelling the optimization-based schedule generator of a generic production scheduling system. *Int. J. Prod. Res.*, **45**(8): 1763–1789.

[17] Ghaelib, M., P.A. Bahria, P. Leec, et al. 2005. Petri-net based formulation and algorithm for short-term scheduling of batch plants. *Comput. Chem. Eng.*, **29**: 249–259.

[18] Silva, M. and R. Valette. 1989. Petri nets and flexible manufacturing. In *Advances in Petri Nets* (*Lecture Notes in Computer Science,* Vol. 424), Springer, Berlin, pp. 374–417.

[19] Zimmermann, A., K. Dalkowski, and G. Hommel. 1996. A case study in modeling and performance evaluation of manufacturing systems using colored Petri nets. In *Proceedings of the 8th European Simulation Symposium,* Genoa, Italy, pp. 282–286.

[20] Piera, M.A., M.E. Narciso and R. Buil. 2009. Flexible manufacturing systems. In Y. Merkuryev et al. (eds), *Simulation-based Case Studies in Logistics: Education and Applied Research,* Springer, London, pp. 109–126.

[21] Kristensen, L.M. and M. Westergaard. 2007. The ASCoVeCo state space analysis platform: next generation of tool support for state space analysis. In *Proceedings of the 8th Workshop on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus,

Denmark, pp. 1–6. http://www.daimi.au.dk/CPnets/workshop07/cpn/papers/Tutorial01.pdf.

[22] Srinivasan, R. and V. Venkatasubramanian. 1998. Automating HAZOP analysis of batch chemical plants: Part I. The knowledge representation framework. *Comput. Chem. Eng.*, **22**(9): 1345–1355.

[23] Moore, K.E. and S.M. Gupta. 1995. Stochastic colored Petri net models of flexible manufacturing systems: material handling systems and machining. *Comput. Ind. Eng.*, **29**(1–4): 333–337.

[24] Chincholkar, K. and O.V. Krishnaiah Chetty. 1996. Stochastic coloured Petri nets for modelling and evaluation, and heuristic rule base for scheduling of FMS. *Int. J. Adv. Manuf. Technol.*, **12**: 339–348.

[25] Mitchell, B., L.M. Kristensen and L. Zhang. 2007. Formal specification and state space analysis of an operational planning process. *Int. J. Softw. Tools Technol. Trans.*, **9**(3–4): 255–267.

[26] Reyes, A., H. Yu, G. Kelleher, et al. 2002. Integrating Petri nets and hybrid heuristic search for the scheduling of FMS. *Comput. Ind.*, **47**(1): 123–138.

[27] Yu, H., A. Reyes, S. Cang, et al. 2003. Combined Petri net modelling and AI-based heuristic hybrid search for flexible manufacturing systems-part I: Petri net modelling and heuristic search. *Comput. Ind. Eng.*, **44**(4): 527–543.

[28] Yu, H., A. Reyes, S. Cang, et al. 2003. Combined Petri net modelling and AI-based heuristic hybrid search for flexible manufacturing systems-part II: Heuristic hybrid search. *Comput. Ind. Eng.*, **44**(4): 545–566.

[29] Kristensen, L.M., P. Mechlenborg, L. Zhang, et al. 2008. Model-based development of a course of action scheduling tool. *Int. J. Softw. Tools Technol. Trans.*, **10**(1): 5–14.

[30] Piera, M.A., M. Narciso, A. Guasch, et al. 2004. Optimization of logistic and manufacturing systems through simulation: a colored Petri net-based methodology. *Simul-T Soc. Mod. Sim.*, **80**(3): 121–130.

[31] Konar, A. 2000. *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*. CRC Press, Boca Raton, FL.

[32] Russell, S. and P. Norvig. 2003. *Artificial Intelligence: A Modern Approach*, 2nd edn, Prentice Hall, Englewood Cliffs, NJ.

[33] Narciso Farias, M.E., M.A. Piera Eroles and J. Figueras. 2005. Optimización de Sistemas Logísticos Mediante Simulación: Una Metodología Basada en Redes de Petri Coloreadas. *Rev. Iberoam. Autom. In.*, **2**(4): 54–65.

[34] Fisher H. and G.L. Thompson. 1963. Probabilistic learning combinations of local job-shop scheduling rules. In J.F. Muth and G.L. Thompson (eds), *Industrial Scheduling,* Prentice-Hall, Englewood Cliffs, NJ, pp. 225–251.

[35] Dauzère-Péres S. and J.B. Lasserre. 1994. An integrated approach in production planning and scheduling. *Lect. Notes Econ. Math.*, **411**: 114–126.

[36] Park, B.J., H.R. Choi and H.S. Kim. 2003. A hybrid genetic algorithm for the job shop scheduling problems. *Comput. Ind. Eng.*, **45**(4): 597–613.

[37] Udomsakdigool, A. and V. Kachitvichyanukul. Heterogenous ant algorithm for job shop scheduling. In *Proceedings of the 2005 International Conference on Simulation and Modeling*, Bangkok, Thailand, pp. 120–125.

[38] Goncalves, J. F., J. J. de Magalhaes Mendes and M.G.C. Resende. 2005. A hybrid genetic algorithm for the job shop scheduling problem. *Eur. J. Oper. Res.*, **167**(1): 77–95.

[39] Kun Tu, Zhifeng Hao and Ming Chen. 2006. PSO with improved strategy and topology for job shop scheduling. In *Advances in Natural Computation* (*Lecture Notes in Computer Science,* Vol. 4222), Springer, Berlin, pp. 146–155.

[40] Senthil Velmurugan, P. and V. Selladurai. 2007. A Tabu search algorithm for job shop scheduling problem with industrial scheduling case study. *Int. J. Soft Comput.*, **2**(4): 531–537.

[41] Pongchairerks, P. 2009. Particle swarm optimization algorithm applied to scheduling problems. *ScienceAsia*, **35**: 89–94.

[42] Beasley, J.E. 2009. *OR-Library*. http://people.brunel.ac.uk/∼mastjjb/jeb/info.html.

[43] Narciso, M.E., M.A. Piera and J.-C. Hennet. 2003. Generation de Plan par Exploration Selective d'un Arbre de Couverture. In *Proceedings of 4e Conférence Francophone de Modélisation et Simulation*, Toulouse, France.

[44] Narciso, M.E., M.A. Piera, and A. Guasch. 2003. A knowledge representation approach suitable to support heuristic search methods. *Fr. Art. Int.*, **100**: 397–408.

[45] Narciso, M.E., M.A. Piera and A. Guasch. 2004. A decision support system to tackle the vehicle routing problem. *Fr. Art. Int.*, **113**: 375–382.

Mercedes Narciso *is a lecturer at the Universitat Autònoma de Barcelona, where she obtained her PhD in 2007. She studied systems engineering at the University of Los Andes, Venezuela. After working several years in a private company (consultancy and development of software in direction positions), she worked as a researcher in the Computer Science Department at the University of Los Andes. In 1998, she received her MSc in computer science from this university. Her research areas include modeling and simulation of industrial processes, airport and air transport systems; software engineering and artificial intelligence. Since 2003, she has been a researcher of LogiSim, a Center for Simulation and Optimization of Logistics Systems.*

Miquel Angel Piera *is an assistant professor at the Universitat Autònoma de Barcelona and during the last 5 years has lead the 'Aeronautical Management' degree, establishing different collaboration agreements with airlines, carriers and airports to improve air transport logistic operations. His research interest focus on logistic systems, causal modeling and discrete event system simulation, he has been very active in the simulation community organizing as General Co-Chair for more than five International Conferences (I3M). At present, he is a co-founder of a spin-off company (DLM) focused on modeling and optimization of complex systems, and the head of the research team LogiSim.*

Antoni Guasch *is a professor at the Universitat Politècnica de Catalunya focusing on modeling, simulation and optimization of dynamic systems, especially continuous and discrete-event simulation of industrial processes. He has lead 43 industrial and scientific projects related with modeling, simulation and optimization of nuclear, textile, transportation, car manufacturing and steel industrial processes. He is currently conducting a research project sponsored by SIEMENS related to the development of power management optimization algorithms for the Barcelona new L9 subway network. He is also contributing to the development of TooPath, a web server system for the free tracking of mobile devices.*