Università della Calabria
Dipartimento di Elettronica, Informatica e Sistemistica

Doctor of Philosophy in Operations Research (MAT/09) – XXI edition

PH.D. DISSERTATION

Modeling, Simulation and Optimization in Logistics

Roberto Trunfio

Supervisor

Prof. Lucio Grandinetti

Advisor

Prof. Pasquale Legato

Academic Year 2007-2008

To Antonella...

Table of Contents

Acknowledgements

My first sincere thanks go to Prof. Pasquale Legato, my supervisor, for his many suggestions and constant support during my doctoral program.

I am grateful to Prof. Roberto Musmanno who gave me the opportunity to work at the NEC Italia S.r.l. *Center for High-Performance Computing and Computational Engineering* (CESIC).

I am also thankful to my PhD colleague Rina Mary Mazza for her friendship and for being so supportive during the three years of PhD studies.

I would like to express my heartfelt appreciation to Daniel Gullì for his loyal friendship and precious technical support.

Of course, I am grateful to my family for their precious advice and *love*.

I am grateful to my family for all the sacrifices they have made for me to succeed throughout my life and for their precious advice, *love* and support.

The foremost thanks go to Antonella: she literally saved my soul in the early years of chaos and confusion and illuminated my path. Without her love this work would never have been completed.

Rende (CS), Italy — Roberto Trunfio
December 1, 2008

Introduction

Nowadays, the competitiveness of a company concerns both the organization and the management of the business processes, from the relationship with suppliers for the supply of raw materials to the relationship with customers for delivery of finished products. In this context, *logistics* has a key role in modern systems for goods manufacturing and freight transportation, and, therefore, the success of a company is related to the optimum management of the logistics.

The need to represent the overall business decisions in a dynamic and uncertain context results in a growing demand for *Research & Development* of Operations Research tools for modeling and simulation of logistics processes, with particular demand for mathematical programming models combined with stochastic simulation tools.

Thus, this thesis is concerned with the definition of OR methodologies for some practical applications in logistics and focuses its attention on methods for integrated representation of the decisions and process in logistics. In particular, the necessity to model in an integrated manner strategic, tactical and operative processes for the definition of a tool for the optimal management of a logistical system, drawn this thesis to highlight the need of modeling and optimizing large and complex logistical systems. In this area, some research efforts have been addressed by using mathematical programming models formulated in a deterministic-static environment. Vice versa, discrete-event simulation models in a stochastic-dynamic environment are well capable of representing the entire processes. Hence, simulation results to be an effective tool for decision making at all decisional levels.

The first Chapter of the dissertation is devoted to the study of modeling paradigm

based on both reductionist and holistic approach and devoted to the representation of logistical processes and the formalization of problems with complex scheduling/assignment constraints. A new modeling paradigm based on the process interaction conceptual framework is also presented. The proposed modeling paradigm is well suited to represent the logistical processes, distinguishing between "atomic" components which, in fact, implement natural resources, and "manager" components who, conversely, incorporating methods of allocation of resources and sequencing of activities. Finally, an example of modeling of logistic processes of the Gioia Tauro maritime container terminal is proposed, with particular attention to the process of discharging/loading of a vessel under a set of complex constraints.

The second Chapter propose an overview of *Simulation-based Optimization*. Therefore, the Chapter is devoted to describe a generic framework for simulation-based optimization and to discuss about the features of the most known optimizer for commercial simulators. Moreover, two frameworks recently developed are introduced. The first framework is the well-known *Nested Partitions* method, while the second is the *Balanced Explorative and Exploitative Search*. The frameworks are deeply discussed and specialized on a problem of production logistics.

The *Quay Crane Scheduling Problem* is taken in the third Chapter, with a mixed integer programming formulation based on position variables and devoted to minimize the *makespan* (i.e., the *vessel overall completion time*). The proposed model is compared with another celebrated mathematical model. Successively, the frameworks proposed within the previous Chapter are here specialized on the quay crane scheduling problem and numerical results on both deterministic and stochastic version of the problem are reported. Instances are finally depicted in the apposite Appendix B.

The thesis concludes with Appendix A. This Appendix completes the modeling effort of the logistical processes described in Chapter 1 by proposing two mathematical formulations for the assignment of the quay cranes to the incoming vessels and the successive deployment of the cranes along the quay (with respect to non-crossing constraints).

Chapter 1

Modeling of Logistic Systems: Processes and Models at a Maritime Container Terminal

Since some decades, discrete simulation has become the most powerful tool in modeling logistic systems in a dynamic stochastic environment. An open challenge is trigged by the need to devise ways of developing easy-to-read and expressive visual modeling paradigms. Most modeling paradigms, as *Event Graphs*, *Petri Nets* and *Hierarchical Control Flow Graphs*, currently adopted mainly in an academic context, have been generally supplanted by simulation packages in real system applications. Nevertheless, our belief is that these are more expressive and powerful in modeling logistic systems. We rely on an innovative visual modeling paradigm based on the process interaction conceptual framework and on the holistic modeling approach, which we are presenting in this paper.

This Chapter focuses on the way of representing processes, resources and entities that compose our simulation modeling paradigm. Here we aim to design a modeling paradigm that is able to provide a clear and detailed description of the logistical processes that arise in a maritime container terminal. Modeling capabilities of our modeling paradigm are compared to those of Event Graphs, Petri Nets and Hierarchical Control Flow Graphs within a real case study.

1.1 Introduction

The increasing international division of labor in the course of globalization, the resulting trade movements and the consequent need for just in time goods production and faster freight transportation, lead manufacturing and transport companies to study and develop efficient and effective planning and control (P&C) tools for decision making, at all decisional levels. Research and development have a crucial role in this business by providing sophisticated decision support systems on powerful and easy-to-use modeling platforms. As stated by several authors [95, 35], in a stochastic dynamic environment, discrete event simulation (DES) models are well capable of representing the behavior of large systems (e.g., manufacturing systems and supply chains). Thus DES models are widely adopted as planning and control tools to estimate system performances under uncertainty and conduct scenario analysis.

Modern, commercial DES simulation packages based on a "point & click" logic – that is what Pidd [65] calls *visual interactive modeling simulators* (VIMSs) – are devoted to minimize the system modeling effort by hiding the behavior of the components adopted to construct the model, but at a loss of model readability, understanding and customization. In opposition, a *modeling paradigm* (MP) is a visual modeling approach based on a formalism designed on a worldview which requires a lot of modeling ability, and which provides superior representational capabilities.

In the past, a lot of interesting and powerful DES modeling paradigms (MPs) have been developed based on the three classical worldviews (or *conceptual frameworks*), i.e. *Event Scheduling* (ES), *Activity Scanning* (AS) and *Process Interaction* (PI) [59]. The most notable MPs developed using these conceptual frameworks are respectively: *Event Graphs* [79], *Petri Nets* [62] and *Hierarchical Control Flow Graphs* [22].

Main conceptual frameworks Derrick, Balci and Nance [21] stated that a worldview is an underlying structure and organization of ideas which constitute the outline and basic frame that guide a modeler in representing a system in the form of a model.

When using the ES conceptual framework, the modeler describes the system of interest in terms of *events* which occur during system simulation. This worldview is

based on a time-ordered schedule of events, called the *event list*, which will occur in the future simulated time. The event list is constructed at runtime, where the events occurrence in the future depends on the length uncertainty of the simultaneous activities contained in the modeled stochastic system. Some events are *determined*, that is they occur at a known future time (e.g., the bootstrapping of arrivals); while others are known as *contingent* or *conditional* and their occurrence depends upon the satisfaction of some sets of conditions not predictable in advance (e.g., the "start service time" depends from an unpredictable customer arrival and is conditioned by server availability). The implementation of the ES worldview is based on the definition of an *event routine* associated to each event. An event routine, according to Pidd [66], is a "set of actions that may follow from a state change in the system". The scheduling and un-scheduling of events and logical checks for contingent events is demanded to an event routine. The ES is the widespread worldview, seeing that it stands out for lowest burden on executive. Otherwise, it has a high burden on modeler, a very low maintainability, a poor natural representation capability and a high effort required for the time of development, which makes the MPs developed using this conceptual framework not appealing for large and complex system modeling.

In the AS worldview, the modeler is required to identify all the system objects that must be modeled, and then the *activities* that the objects perform, together at the conditions under these activities take place. An activity is initiated by a state change determined by a test set of boolean conditions, called *testhead*. Testheads are used to link the various activities and to produce the state transitions of the model objects and the intersections among them. Thus, a model is composed by a set of testheads and the associated resulting action awaiting execution at the appropriate time. AS worldview implementation is based on a two-phase monitor which performs a *time scan* (a routine that advances the simulated time) and an *activity scan* (a routine which determines the next executed activity checking all the testheads). In a different way by the ES worldview, the AS conceptual framework is affected by a high burden on executive, but it has a low burden on modeler, a high maintainability, a good natural representation capability and a low effort required for the time of development.

The PI conceptual framework better suits the needs for model readability and understanding. In fact, while ES and AS modeling approaches are based respectively on events and activities, the PI is based on the concept of *process*. A process is a complex concept which represents a flow of events and activities through which a particular model object moves; therefore, in a model specification, it describes the life cycle of a model object. Whilst a model object moves through its process, it may experience certain delays and be hold in its movement. Thus, the first thing to do in the PI worldview is to identify all the model objects which are involved in the model specification. Subsequently, the modeler must specify the sequence of events and activities for each model object. For this reason, a process is often represented with a schematic representation known as *flow-chart*, where the events and activities that compose the process are the nodes of the chart. A flow-chart is usually represented as a directed graph. For each activity included in a process routine there is a stretch of simulated time (even null). The PI qualities are a moderate burden for the modeler, a high maintainability, an excellent natural representation capability. The most notable drawback is the high burden on execution, even if modern object-oriented approaches better fit the implementation needs of this conceptual framework, and the effort required for the time of development, which is quite high.

Main modeling approaches Modeling a stochastic system using a MP or, in alternative, a commercial VIMS (e.g., GPSS and Arena) implies the use of a specific modeling approach. As anticipated before, the two classical dichotomous modeling approaches are *reductionism* and *holism*.

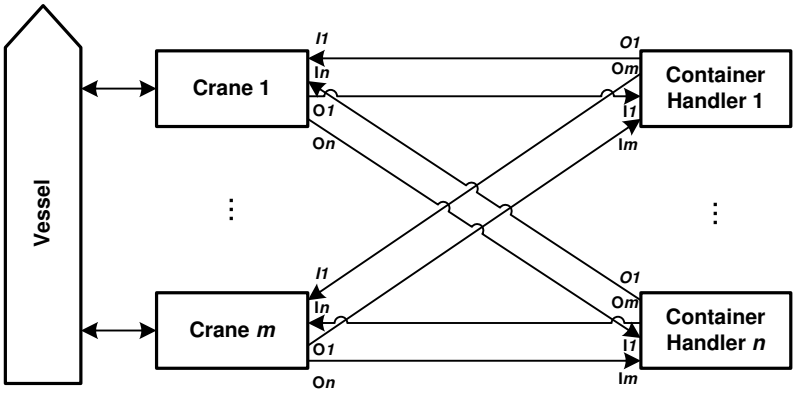
Reductionism relies on the belief that a large and complex system may be decomposed into its constituent parts without any loss of information, predictive power or meaning [67]. The totality of MPs and VIMS are developed using the *modular modeling* approach, which is the most notable expression of reductionism. In fact, a modular model, as argued by Zeigler [96, 97], must satisfy the following conditions: *i*) each module (or component) must not directly access, modify or refer to the state of any other module; and *ii*) each module must have a known input and output ports through which all interaction with the exterior is pursued. Therefore, by mean of the modular approach, a system is decomposed into its constituent parts

or components, depicted as black boxes, as requested by the reductionism.

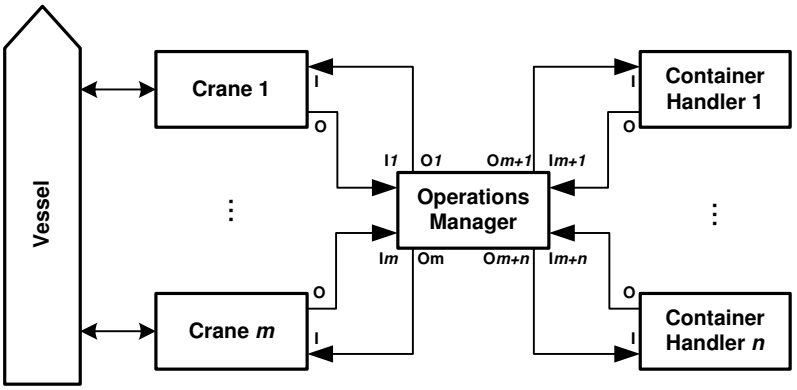
Unfortunately reductionism does not consider that large and complex logistic systems cannot be decomposed into their constituent parts with the guarantee of all the above conditions. This statement can be practically proved by trying to analyze real systems, e.g. *maritime container terminals*. In this context, it is easy to verify that it is not possible to decompose the system into its constituent parts (i.e. *logistic processes* – see Stahlbock and Voß [86] for an up-to-date review on logistic processes in maritime container terminals) and analyze them separately, because they are partially or entirely related: some significant loss of information, predictive power and meaning will occur. According to Pidd and Castro [67] we do believe that the best approach for the management of a large and complex dynamic discrete system should be based on holism.

Holism assumes that systems possess some properties that are meaningful only in the context of the whole and not in the parts. This is just what occurs in logistic systems, due to the fundamental role of logistics which amounts to integrate manufacturing and transportation processes as well as to coordinate the usage of shared resources. Holism allows to develop a simulation model by means of many *atomic components*, and using few context-specific components, that are called *manager components*. Atomic components are characterized by a simple behavior and are un-dependent by the model layout (as in the reductionist approach). Manager components have a complex behavior (e.g., they may use complex algorithms to make resource assignment or task scheduling) and they are designed around a specific system layout, therefore each of these components has a high-level view of the system and of its components. Atomic components are usually linked to one or more manager, which act as a hub. This approach it may remarkably reduces the overall number of links in the model, improving the model readability.

To show the difference between the two modeling approaches, here we provide an introductive example of a model of some logistical processes of a typical maritime container terminal. In Section 1.6 is provided a more detailed description of a holistic simulation model of a real container terminal.



(a) Reductionistic approach.



(b) Holistic approach.

Figure 1.1: Vessel discharging/loading model.

We are interested in depicting the model for the process of vessel discharging/loading through m *quay cranes* (QCs) and simultaneously transferring containers from the ship at the storage area (and *vice versa*) by means of a fleet of n *shuttle vehicles* (SVs). In Figure 1.1(a) is depicted the corresponding model using the reductionist approach. In this case, we have m buffered QCs ("Crane" component), used to perform containers discharge and loading operations of a specific vessel. These cranes interact with n resources for container handling ("Container Handler" component), i.e., special SVs that are used to transfer, set-up and set-down containers and are known as straddle carriers (SCs). Both component types interact by message passing in order to negotiate containers transfer. Each Crane

component interacts with the static component vessel, with the purpose to negotiate the access of the assigned tasks (i.e. decks and holds that must be discharged/loaded under specific constraints): then, the end-user must specify for each Crane the list of tasks that must be performed. Another parameter that must be specified is the assignment at one specific Crane of each Container Handler (runtime assignment of the Container Handler is difficult with this approach). Besides, with this approach we have $4 \cdot m \cdot n$ links among components.

The corresponding model designed using the holistic approach is proposed in Figure 1.1(b). This model introduce a manager component (the "Operations Manager") to coordinate the assignment of the vessel's tasks to the each Crane component and the integrated management of the Container Handler components within the transfer process of containers forth and back from the quay to the yard. In particular the routing function of straddle carriers among several container storage points on the yard and vessel berthing points on the quay is recognized as crucial with respect to the whole performance of the terminal and the Operations Manager is a component that can include several policy of resource allocation. In this model, the Crane components and the Container Handler components interact via the manager component, which act as an interface between both component types. Obviously, in this case the end-user must specify key parameters only for the Operations Manager and not for all the others $m + n$ resource components. The overall number of links among components is also reduced to $2(m + n)$: in real cases we have about 4 QCs per vessel and 4 SCs per QC, which means 256 links using the reductionist approach versus 40 using the holistic approach.

It should be clear at this point that the use of a holistic approach to model a whole system achieves better results in terms of the replication of the real system behavior and model readability. Thus, developing an MP or VIMS by using the holistic approach should be appropriate. However, considering that both alternatives are affected by an intrinsic difficulty in model understanding, which makes simulation models unpopular at the model end-users, here we propose an MP which provides: *i*) a high representational capability from the conceptual point of view, as well as *ii*) a common language for both modelers and end users from the model understanding sight.

The definition of an innovative MP for logistic systems operating under uncertainty based on a holistic approach, is the basis for developing a simulation-based optimization platform. Besides, the considerations about strong and weak points of the ES, AS and PI worldviews convince us that we must define our MP using the PI conceptual framework. Afterwards, in the next sections we introduce the *Event Graph* (EG), *Petri Nets* (PN) and *Hierarchical Control Flow Graph* (HCFG) simulation models, to describe three successful modeling approaches based on the three classical conceptual frameworks from which we define our MP. Thus, we dedicate the subsequent section describing our MP, inspired by a previous work [47].

Successively, we introduce problems and features of a maritime container terminal, proposing a high-level simulation model of a real container port. Then, to compare our MP with three different and successful MPs (i.e., EG, PN and HCFG), the attention is focused on a part of the proposed simulation model and the part is modeled using the MPs introduced in this Chapter. Thus, different way of modeling the same reality are shown.

After that, we discuss some specific issues when developing a friendly tool for the integration of optimization techniques within simulation platforms. Conclusions are finally provided.

1.2 Event Graph

The EG is a MP based on ES worldview developed by Schruben [79]. An EG is an oriented graph, where a vertex represents an event an arcs stands for relationship among events. An EG model may be disconnected, thus it may consists of a set of components whose behavior is an EG itself. In this case, the components interaction is based on underlying model variables. Besides, EG allows hierarchical modeling regrouping sub-graphs in a single vertex and exploding the compound vertex if needed. Therefore, EG is a powerful and quite simple tool for discrete event system modeling.

The following subsection provides a detailed description of the EG based modeling.

1.2.1 Elements and Structure

Modeling through EG requires to define and number all the events of the system, therefore one has to identify all the prominent system state changes (i.e. a variation in the system state variables) which take place when events occur. All the events are depicted as a circle, or rather as a vertex of the graph. The occurrence of the event i modifies a definite set of system state variables, whether deterministic or stochastic. The use of these set of state variables is needed to implicitly represent system entities (jobs) – e.g. increasing a variable which represents the number of queuing entities [93]. Events are provided of parameters (e.g., the *resource id* for the *start service* event in $M/M/m$ queuing models), therefore a vector of event parameters must be provided. The use of vertex parameters minimizes the model overall dimension, in fact events which differ only by parameters value are depicted through the same vertex. Some practical examples about system modeling using EG graphical modeling can be found in [77, 80].

A pair of events A and B is connected by a directed edge (arcs), so an edge point outs how the event A (the edge head) influences the occurrence of the event B (the edge tail). In the EG paradigm there are two types of edges: *scheduling edge* and *cancelling edge*. The first type is depicted as a solid arc, whereas the latter as a dashed arc. Similar to events, edges have a set of attributes representing a set of logical and temporal expressions (i.e., *delay times*). Logical expressions may be constructed coupling simple conditions using both i) boolean operators *and* ($\vee$), *or* ($\wedge$) and *not* ($\neg$) and ii) relational operators ($<$, $\leqslant$, $=$,$\neq$, $\geqslant$, $>$). Logical and temporal expressions may be based on values generated from random variables sampled using such distribution by means of the convenient software library [38]. If necessary, the edge head and tail can be the same vertex; in addition, more arcs can exist between the same pair of nodes. Moreover, if an edge is not marked by any attributes, (scheduling or cancelling) it is called *unconditional* edge.

According to Schruben, events may execute the following actions: i) transformation of state variables, ii) generation of edge delay times, iii) testing of event logical conditions, and iv) scheduling/cancelling future events.

In the EG based modeling, simulation clock and future events list are not explicitly represented, even if in a subsequent work Yücesan and Schruben [93] introduced

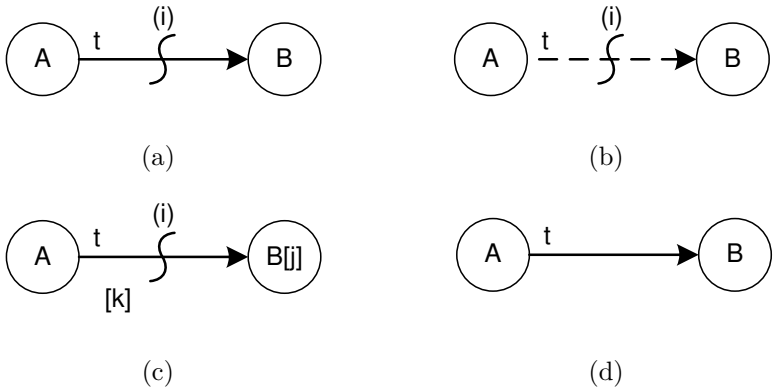


Figure 1.2: Event Graph notation.

current simulation clock as a model state variable. In that way, EG allows designing a set of variable to collect system performance measures. Schruben remarks that the use of this class of variables as edge conditions is not allowed.

In Figure 1.1 the constructs adopted to model a system using EG are depicted. Through the formalism in Figure 1.2(a) is possible to model the scheduling of the event B after t delay time by the occurrence of the event A, only if the logical condition i holds. Likewise, in Figure Figure 1.2(b) the modeling case for event cancellation is illustrated. Finally, Figure 1.2(c) shows the notation for the scheduling of the event B as described in case (a), with the parameter vector j assuming the current values of the vector k. For sake of completeness, Schruben claimed in his first work about EG that cancelling edges are useful, but not absolutely necessary. Unconditioned edges are depicted as a simple directed edge, as in Figure 1.2(d). An unconditioned edge depicts a relation among two events A and B that is not bounded by any timed or boolean condition.

Schruben developed a set of rules that must be fulfilled to attempt a right model simulation. The most notable rules regard the i) event initialization, and ii) event execution priority.

The former rule concerns the set of events that must be scheduled to prevent simulation starvation. In each EG strongly connected events can be identified, by the subgraph obtained once all the cancelling edges have been removed from the EG. Therefore, the rule states that "at least one event in each strongly connected component of an event scheduling subgraph with no entering edges needs to be

initially scheduled in a simulation program".

The latter rule has been developed to avoid simulation deadlock when multiple events are scheduled at the same simulated time. In this case, an execution priority must be decided. Thus, the rule suggests "to establish a priority between the events A and B if there is a non-null intersection between the set of state variables possibly altered by event B and the set of state variables involved in the conditions of all the edges originating by vertex A". As it is easy to recognizes, the application of the latter rule makes modeling through EG not as easy as it could seem.

1.3 Petri Nets

A Petri Net (PN) is a graphical and mathematical modeling tool developed by Petri [62] in his PhD thesis. PNs have a simple graph-based representation. A PN has two components: a *net* and *initial marking*. The net is a directed graph with two sorts of nodes, called *places* and *transitions*, such that there is no arc between two nodes of the same sort (sometimes are used different arc type, e.g. *inhibitor arcs*). Places are graphically represented by a circle while a transition by a bar, as depicted in Figure 1.3. Each arc has a weight that is not displayed in case of value equal to one.

Figure 1.3: Petri Nets notation. On the left the classic Petri Net components, on the right an example of Petri Net with an inhibitor arc.

A place can store a natural number of *tokens*, represented by black dots. A *marking* is a deployment of tokens among places and it corresponds to a state of the PN. Petri calls *input places* those places which are linked through transition incoming arcs (or *input arcs*), whereas he calls *output places* those places which are corrected via the outgoing arcs of the transition (or *output arcs*). A transition acts on input tokens by a process called *firing*, and it may fire whenever it is enabled. A

transition is enabled when each input place has at least a number of tokens equal to the weight of the arc. A transition may fire when it is enabled; its firing changes the marking of the net consuming a number of tokens (equal to the weight of each input arc) from each of its input place, and producing an amount of tokens (equal to the weight of each output arc) to each of its output place. Transition does this atomically, in one non-interruptible step. Execution of PNs is non-deterministic, therefore: *i*) multiple transitions can be enabled at the same time, any one of which can fire; *ii*) none are required to fire, i.e. they fire at will, between time 0 and infinity, or nothing fires at all. Transitions are in conflict if they share input places, therefore in this case a firing priority must be declared.

Figure 1.4 depicts an example of a PN before and after firing of a transition.

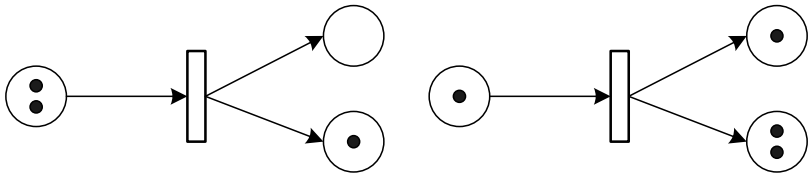


Figure 1.4: Petri Nets: Before firing (on the left side) and after firing (on the right side).

A transition with an input inhibitor arc is enabled when both the following conditions occur: *i*) each of all input places connected to the transition via normal arcs have a number of tokens at least equal to the weight of corresponding arcs; *ii*) each of all input places connected by mean of inhibitor arcs have no tokens.

A lot of properties mark a PN. *Liveness* is the first property. A PN is live if every transition can always occur again. In particular, if for every reachable marking (i.e., every marking which can be obtained from the initial marking by successive occurrence of transitions) and every transition t it is possible to reach a marking that enables t. Moreover, a PN has the property to be *deadlock-free*, that it is a weaker property than liveness. A PN is deadlock-free if every reachable marking enables some transitions.

A PN is *bounded* if exist a number b such that no reachable marking puts more than b tokens in any places. Places in a PN are often used to model buffers and

registers for storage data (while transitions represents activities). If the PN is unbounded, then overflows can occur in these buffers/registers.

Another important property regards the reachability of the initial marking. In PNs, a marking is a *home marking* if it is reachable from every reachable marking. Thus, if the initial marking of a PN is a home marking then the PN has the property to be *cyclic*.

Many extensions of the original PN model have been proposed through the decades: *Coloured Petri Nets* (CPNs) have been developed in order to distinguish different token types [31]; *Time Petri Nets* (TPNs) have been developed in order to include the concept of time. In DES models, a key role is covered by *Stochastic Petri Nets* (SPNs), which are a particular specialization of TPNs. In the following, we introduce TPNs and SPNs.

1.3.1 Timed and Stochastic Petri Nets

The ordinary PNs do not include any concept of time. Therefore, the original PN MP is not able to describe the evolution over the time of dynamic systems. Responding to the need for the temporal performance analysis of discrete-event systems, time has been introduced into PNs in a variety of ways (associating a time delay to places, transitions, arcs or tokens), as with TPNs. A TPN is similar to a PN with the addition of a deterministic firing delay associated to each transition (i.e. a *timed transition*). A delay specifies the time when the firing effects of an enabled transition become evident. Whenever there are both timed and not timed transitions (i.e. *immediate transitions*), transitions that are not timed have a higher priority than timed ones.

However, fixed delays are not appropriate for most real systems characterized by an intrinsic variability. Thus, since most of the variables involved in complex DES systems are stochastic by nature, SPNs have been developed as a stochastic extension of TPNs. In a SPN, a probability distribution is assumed for each time delay (more formally *stochastic delay*) [53]. However, to make analysis tractable typically only a restricted set of probability distributions is allowed. For instance, a particular and widely used SPN developed by Marsan, Balbo and Conte [50], known

as *Generalized Stochastic Petri Net* (GSPN), allow both *immediate transitions*, i.e. transitions with no delay, and *timed transitions*, i.e. transitions with exponential delays.

Some additional considerations about transitions enabling and firing rules are necessary for TPNs. Considering TPNs where delays are determined by timed transitions, a possible approach is based on the so called *race semantics*. In the race semantics, time delay is associated to the enabling time and once a transitions has been enabled it races against each other enabled transitions to fire for first. Another approach is the *preselection semantics*, where time delay is associated to the firing time and enabled transitions are scheduled to fire after a time delay using a priority or probabilistic mechanism. The race semantics approach allow for more compact model representation, while the preselection semantics are more intuitive and easier to use.

1.4 Hierarchical Control Flow Graphs

HCFG models are a MP developed by Fritz and Sargent [22] and based on a modified version of the PI conceptual framework [18] that includes concepts as *encapsulation* and *locality*. HCFG models are a hierarchical extension of the *Control Flow Graph* (CFG) models formerly designed by Cota and Sargent [17]. Modeling by CFG models means to specify the behavior of each model component, i.e. a *process*, using a graph called CFG. Processes interact via message passing using *input* and *output* ports of the components that are connected through *channels* (directed edges). Therefore, each component is depicted as a box equipped with a set of labelled input ports and output ports. A component has a *type* and an *instance name*. This distinction is necessary to avoid misunderstanding in a simulation model; in fact, a component type should be instanced many times in the same model and the instance name works as a unique *ID* for components of such a type in the same layer. The component inside (*inner view*) depicts the component behavior, while the component outside (*outer view*) shows the relationship between model components linked together by means of directed edges forming a net. The approach that Cota and Sargent used to describe the interaction between model components has inspired

some of those used in modern VIMS, which is one success factor for commercial VIMS.

In the following a comprehensive description of the HCFG models is provided.

1.4.1 Fundamental Elements and Structure

HCFG models are a hierarchical extension of CFG models. Fundamental in HCFG understanding is the introduction of CFG models. A CFG model is composed by a set of components interacting by message passing through *input/output ports* linked solely by mean of directed edges called *channels*. A structure known as *Interconnection Graph* (IG) is used to specify the components links: it is a directed graph in which vertices are model components and arcs are channels.

A channel is able to connect only one output port to an input port and to transfer only one message type, therefore many channels should be instanced between a couple of components. Messages are queued into the input port of a component until the component does not accept them. In the CFG models, system entities are symbolized as messages. Components apply a timestamp to each sent message to show its sending time.

CFG models are not an *easy-to-use* MP for large and complex systems, because they are not designed for hierarchical modeling. This lack has been filled by HCFG models introducing two independent but complementary type of hierarchical model specification. The first is called *Hierarchical Interconnection Graph* (HIG); while the second is know as HCFG.

A HIG is based on the main idea of HCFG models: components should be linked together to compose a new model component (hierarchical approach). Thus, likewise an IG, a HIG is a graphical tool used to represent a hierarchical model depicting components as nodes and channels as arcs. A compound component (a node) should be expanded in order to show the connections between sub-components from its inner view and components from its outer view. Only one HIG must be defined for each HCFG model. Components that are not decomposable are known as *Atomic Components* (ACs), while components resulting by coupling other components are called *Compound Components* (CCs). Therefore, in a HIG un-expandable nodes are

ACs.

As stated before, the outer view of a component depicts the relationship of a component with the others, i.e. a set of input and/or output ports is depicted and directed edges are used to couple components through the ports. In Figure 1.5 is shown the outer view of a component developed using HCFG models.

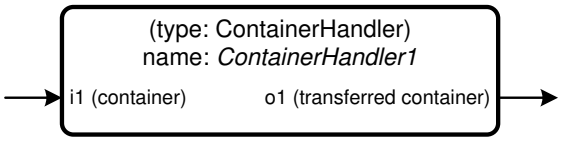


Figure 1.5: Outer view of a component for a container terminal system using HCFG models.

Using HCFG models, we have two different way of representing the inner view of a component. The inner view of a CC is defined using a structure called *Coupled Component Specification* (CCS). A CCS is a directed graph whose nodes are model components and arcs are channels. In Figure 1.6 is depicted the inner view of the component shown in Figure 1.5. As it easy to see, the CCS of a CC does not include the specification of the sub-components behavior. The CCs cover a key role in the modeling of large and complex models, since they allow the modeler to easily design a hierarchical model through components composition and decomposition. In that way, expanding a node in a HIG means to replace the expanded node with the corresponding CCS.

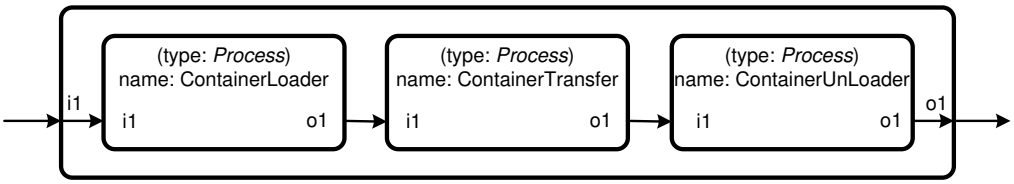
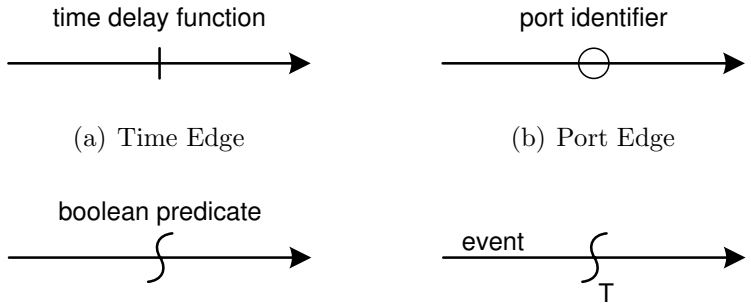


Figure 1.6: Inner view of a component for a container terminal system using HCFG models.

The inner view of an AC is depicted through a HCFG. A HCFG is a tool for the design of components behavior, i.e. it is a hierarchical version of a CFG. A CFG

specifies a component's state space through an augmented directed graph, where a state is depicted as a node and the possible state transitions are shown as directed edges among couples of node. Considering that a process should be suspended when it possesses the thread of control and then it may be reactivated [20, 95], all the states are possible *process reactivation points* (i.e., once a process has been reactivated, its thread of control starts again from the last visited state). Each AC has a set of local variables not visible by other components; these variables comprise a local simulation clock. Three attributes are associated to each directed edge: a *condition*, that defines when the arc is candidate to be traversed; a *priority*, that is used to decide which arc must be traversed when multiple edges are candidate for traversal at the same simulated time; and an *event*, which specifies the state transition that occurs during simulation whenever the arc is traversed.



(a) Time Edge (b) Port Edge

(c) Bool Edge (d) True Edge

Figure 1.7: Edge notation in a CFG.

Edge conditions may be of the following types: *time-delay*, *non-empty input port* and *boolean expression*. Thus, we may refer respectively to *TimeEdges*, *PortEdges* and *BoolEdges*. A TimeEdge is equipped of a time delay function which returns a non-negative number representing the *edge traversal time*, i.e. how many simulated time from the current system clock the edge should be traversed. A PortEdge is an edge associated to an AC input port (many PortEdges can be associated to the same input port). For this edge type, the edge traversal is permitted when at least an un-received message is waiting into the *message queue* of the associated input port. A BoolEdge has a boolean expression which refer only to local variables. The

boolean expression must be evaluated to decide if the edge can be traversed. A subtype of BoolEdge is called *TrueEdge*, that is a BoolEdge whose condition is defined to always evaluate to true. In Figure 1.7 is depicted the notation for all the edge types.

The event is an edge attribute that specifies a set of actions that must be executed when the edge is traversed. Allowed actions include changing in AC local variables, message sending to output ports, and only for PortEdges the message receiving from the associated input port. A *null event* occurs if no action must be performed during edge traversal (in this case the event is depicted with the function "*e-null*", but we prefer to omit it to ease the model readability).

Simulation execution is generally initialized by inserting messages into components input ports. Fritz and Sargent proposed a generic algorithm for model simulation, specifying how to identify the next traversed edge, when advancing simulation clock and how transfer the control between component states.

In a similar way as Schruben proposed for the EG models, whenever using ACs is necessary to design k instances of the same element (e.g. ports and edges), is possible to use *multi-ports* and *multi-edges* recurring to a vectorial notation [78, 37]. Multi-edges are depicted using a dashed line and the edge cardinality is shown in square brackets upon the edge.

In Figure 1.8 is shown an example of CFG for a *SimpleServer* AC as proposed by Sargent [78]. The behavior of this AC is described using the states "idle" and "busy", designed respectively using the vertices "I" and "B". Message arrival from the input port i1 represents the arrival of a job requiring processing. Whenever a message arrival takes place at the same time that the AC status is on the vertex B, then the message is queued in the message queue of the input port i1.

Modeling the behavior of ACs is straightforward in simple systems, but can become really complicated in large and complex systems. Therefore, Fritz and Sargent developed a hierarchical extension of CFG, called HCFG, which guarantees reusability and modularity. The behavior specification of an AC using HCFG is based on the idea that it should be decomposed in partial disjoint behaviors known as *Macro Control States* (MCSs). In that way, it is possible to depict the inner view of an AC by coupling MCSs (likewise for the inner view of a CC). A MCS is a

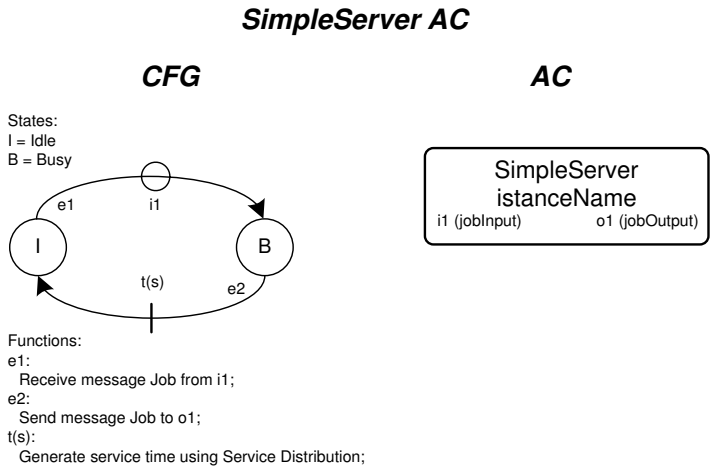


Figure 1.8: The CFG for the AC SimpleServer (left side) and the outer view of the AC SimpleServer (right side).

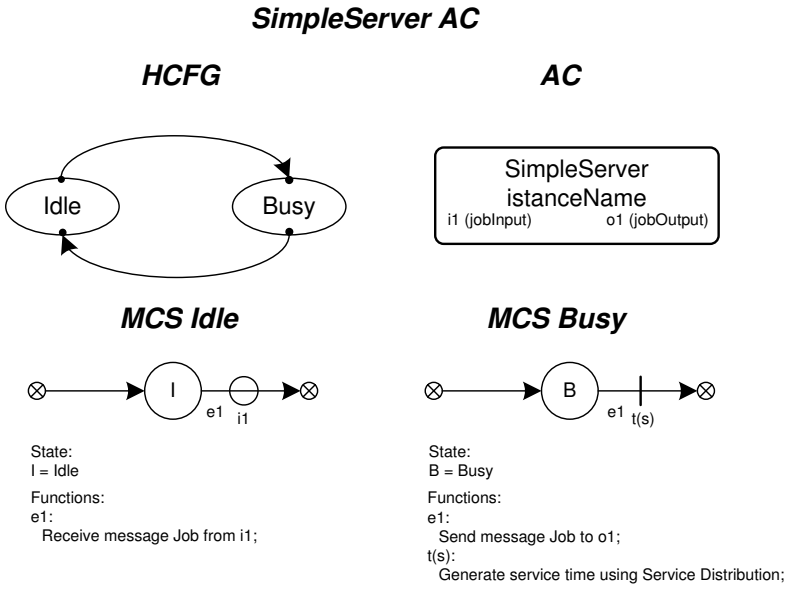


Figure 1.9: A HCFG for the AC SimpleServer.

CFG where nodes are states or MCSs, the input port and output port are explicitly depicted as pins. Thus, the simplest MCS is designed as a CFG. An MCS has no visibility of its outer view; then if a MCS or an AC encapsulates a MCS, its

local variables can not be accessed by the encapsulated MCS. The HCFG of the AC SimpleServer depicted in Figure 1.8 is given in Figure 1.9: this HCFG is composed of two MCSs, the MCS "Idle" and "Busy". Comparing the inner view of the AC SimpleServer provided by the CFG in Figure 1.8 with the one described by the two MCSs of the HCFG in Figure 1.9, it is easy to recognize that the use of HCFG also improves the readability of the behaviour of an AC.

1.5 Holistic Discrete Event Simulation Models with Process Interaction Modeling

Here we describe the MP called *Holistic Modular Process* (HMP) for developing simulation models originally proposed by Legato, Gullì and Trunfio [40]. A HMP model is a holistic MP for DES modeling based upon the PI worldview. Main concepts are presented in the following and its potentiality is illustrated by modeling a typical logistic process that arises in a maritime container terminal. Comparisons with the EG, PN and HCFG models are also provided.

1.5.1 Holistic Modular Process Simulation Models

The MP proposed in this paper is aimed to be flexible and expressive in the modeling of complex systems. It tries to achieve three primary objectives: model readability, reusability and personalization.

Model readability is a property which allows a model to be simple-to-read for a non-modeler. This property has a special importance when top managers are directly involved in scenario analysis: in our MP readability is achieved by describing the components' behavior within a simulation model by a sort of flow-chart. As for re-usage property, our experience at the Gioia Tauro Container Terminal confirms the requirement that a specialized simulation tool has to be reused in some of its forms (model reuse, component reuse, function reuse and code scavenging [64]). Model reuse, under calibration and repeated tuning, occurs as soon as traffic conditions change over time. Furthermore, component and function reuse are both required to

give the operational manager the possibility of quickly implementing a first order model of some emerging situations, before the structured intervention of external expertise. According to our concept of holistic MP, we provide model reusability by means of hierarchical, modular model definition and redefinition of simulation parameters.

Model personalization is the base of an MP for the effective modeling of complex systems. It relies on the user-definition of process properties that allow describing uncommon situations, as it is the case when the modeler is asked to represent local, best practices in logistics organization and management.

An outline of the HMP simulation models follows now. An HMP model includes a set of *model objects*, or *objects* for short. For each model object an inner and outer view is defined. The outer view is depicted as a box equipped of input and output ports (see Figure 1.10). The inner view depicts a sequence of activities and events. The sequence is also called the *model object process routine*, or simply *process*.

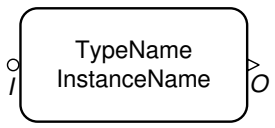
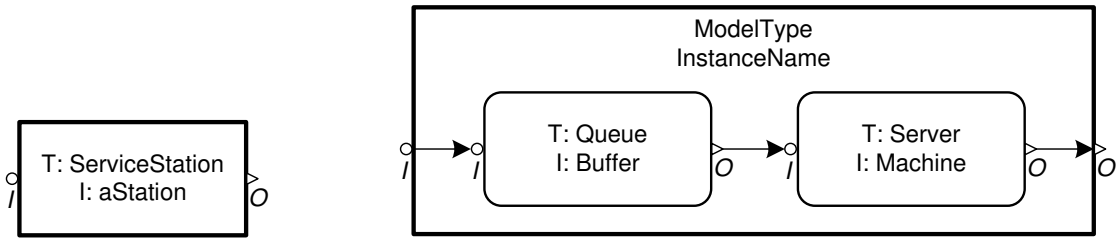


Figure 1.10: The outer view of a model object in a HMP simulation model.

In a similar way as described for HCFG models, relationships between model objects are depicted using directed edges (*channels*). Model objects are equipped with *input* and *output ports* (depicted as shown in Figure 1.10) that can, eventually, be renamed to improve model readability. Two model objects are linked by means of only one channel from an output port of the first object to an input port of the second object and vice versa, i.e. no more than two channels connect directly two model objects. Two model objects interact by message passing via channel, despite a channel has its own direction (the head of the edge is connected to an output port, while the tail to an input port). Multiple types of messages flow forward and backward along a channel, whilst *entities* (or *jobs*) of the simulation model can flow only through the proper channel direction.

In this way, by adopting a modular approach, an HMP simulation model is a net



(a) Outer view. (b) Inner view.

Figure 1.11: A sub-model depicting a service station, composed by a buffer and a server.

of model objects. Therefore, hierarchical modeling is pursued by coupling different processes and grouping the resulting net of model objects into a *sub-model* (which is depicted in a similar to a model object, as proposed in Figure 1.11). In this case, the sub-model can be used to be coupled together with other model objects or sub-models. A requirement when constructing sub-models is that at least one input or output port must be defined and linked to the inner model-objects, otherwise, the depicted model is a *super-model* or *final model*. The inner view of a sub-model is just another HMP model where some channels link the inner part to the outer part of the model through its boundaries.

As stated above, the inner view of a model object is a process, or rather a sequence of activities and events that define the model object behavior. The process is represented as a particular hierarchical flow-chart, called *Event-Activity Diagram* (EAD), where activities and events are nodes and edges fix the logical and temporal sequence between nodes (other node types will be introduced in the following). A process can be constructed hierarchically by grouping events and activities to compose a sub-process. Thus, a sub-process is an EAD itself. By means of a sub-process we provide component reuse and model readability.

Similar to the conical methodology [55, 21], that is a framework for simulation model development, a model definition and specification using two main tools is provided.

The first tool is the *model hierarchy structure* (MHS). The MHS is a tree that provides a high-level model definition showing hierarchically the sub-models and

model objects that compose a simulation model (see Figure 1.12 for an example).

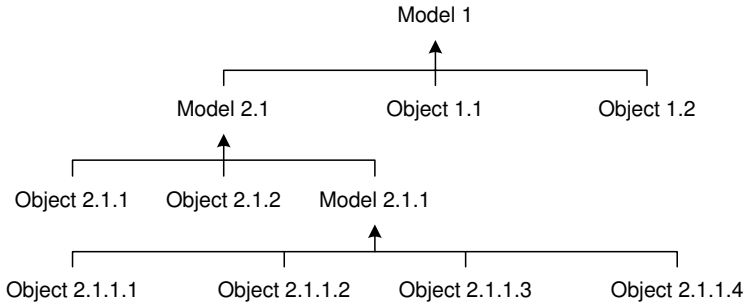


Figure 1.12: A Model Hierarchy Structure (MHS).

A more detailed model definition is obtained by a second tool, the EAD. An EAD provides the process definition. In fact, an EAD defines the structure of a process by means of a sort of flow-chart. Because of the possible use of sub-processes for the process definition, an EAD could be a hierarchical structure (composite nodes are expanded revealing a new EAD) and a hierarchical tree, similar to the MHS, may be used to explore the EAD structure.

1.5.2 Model Objects

Following the PI worldview, the first step consists in identifying all the model objects which are involved in system modeling and detect all common features. Then, class-dependent features are described for the few classes of model objects introduced in our MP. The objects are contextual, so it is necessary to specify the model in which they are defined. The *model name* parameter is used to declare which model an object belongs to. This parameter serves two reasons: first of all, the holistic approach says that such objects can only be used in some contexts; furthermore, the use of sub-models could cause a lot of confusion, especially if a sub-model is exploded (i.e. deleting model boundaries) and the same object is used in a model and in its sub-models. Objects of the same type are identified by the *type name* parameter. If the model name is a parameter that depends on the model, the type name is a non-changeable parameter. Each instance of a certain type of object is

also identified by the *instance name* parameter. The set composed by these three parameters univocally identifies a model object within an HMP simulation model.

There is at least another important parameter that allows managing heterogeneous objects, known as *category name*. The use of the category name allows us to make associations between objects that are apparently disjoined. A possible use can be seen in the development of a simulation package, in the packages for statistical analysis and optimization of system performance measures. As a matter of fact, in this context generic rules and algorithms can be defined over a class of mixed objects - the benefits that can be derived by this approach has been investigated in a companion paper [48].

Model objects have a set of variables and data structures used to support the logical representation of the process behavior. These properties are not explicitly depicted and refer to the code implementation of each model object type. Model objects variables and data structures are accessed by a set of public functions which allow their manipulation. A function is called by message passing, or rather sending an explicit message to call a specific model object function.

Model objects are illustrated in detail in the following.

There are two basic classes of objects: *resources* and *resource managers*. As stated above, entities of the simulation model are depicted as messages that are able to envelop properties, data structures and other entities (e.g., a ship that carries thousands of containers loaded in different holds).

Resources are *active* or *passive* depending on their role in the simulation model. Passive resources are not depicted explicitly and are not able to execute action/events or process entities. Nevertheless, a passive resource is able to execute incoming requests and actions of other model objects (as declared above, a passive resource shows a set of public functions that can be used to manipulate the resource). Passive resources are generally managed by an active resource or a resource manager. Whenever a passive resource is managed by a resource manager, active resources linked to the resource manager are able to overwork it only under the conditions specified by the resource manager. A passive resource must be managed by a resource manager if more than one resource may request it for use during a simulation (e.g., items storage into a shelf using forklifts). An active resource can possess

passive resources and it can offer a service to one or more entities per time. It can also make queries to other objects to which it is linked by message passing via input/output ports. The behavior of an active resource is described using an EAD.

By means of resources one can only represent just a system governed by a few simple rules. As matter of fact, the need of modeling complex systems in a holistic approach leads us to introduce the resource managers. A resource manager is a high-level object, which is able to interact with a set of model objects (also heterogeneous). Resource managers can take decisions (e.g., solve a scheduling or an assignment problem, negotiating the use of a sub-system, etc.) by applying rules and policies and making queries to other model objects. They have free access to modify the behaviour of all the resources in their own model to which they are linked. In a holistic approach, as demonstrated by Pidd and Castro [67], the use of that kind of model objects avoids the explosion of object links and therefore it represents a more easy-to-use modeling tool.

1.5.3 Processes and Event-Activity Diagrams

The role of a process in model object specification is analyzed here and process representation is shown. The interaction of processes during the simulation is briefly described.

According to the definition of process given in the PI worldview overview, a process is a series of temporally related events and activities. Usually flow-charts may be used to represent processes. In our flow-charting methodology, called *Event-Activity Diagram* (EAD), events and activities are nodes, while directed edges define one or more paths that can be covered by a process. Other useful elements compose a process, namely the *logical nodes*.

Activities and events are not intended to perform actions, but to show to non-modelers, in a friendly-way, how a process can work. The role of executing requests, performing actions and introducing time delays between activities and/or events is assigned to directed edges. Therefore, the use of a flow-charting graphical methodology to depict a process allows us to achieve at a good extent the readability objective. Bearing in mind the list of the process components, let us start an in-depth

discussion about these components.

In our MP, activities are classified focusing on the simulation duration of the activity; hence activities are partitioned in *timed* and *instantaneous*. The first type of activities are those able to start operations at a simulated time instant and finish operations in a future simulated time instant. For instance, timed activities are those that perform operations characterized by a variable simulated time length, e.g. waiting or servicing activities. The second type refers to activities that start and end operations at the same simulated time instant, e.g. activities representing a choice or check by a resource or resource manager. Nevertheless, also timed activities can starts and ends operations at the same simulated time instant.

A process needs the specification of an initial activity that is enabled, or rather the activity that possesses the process checkpoint (more details about checkpoints are provided in the following). The initial activity is the activity from which the process starts when it becomes active (initial process state). In an EAD one or more activities per time can be enabled, i.e. when the process flow has been forked in different logical paths. The set of currently enabled activities is called the *process state*.

An event is a fact that forces one out of a set of possible changes of the current state. It may precede or follow an activity, thus representing something that is just happened or that is going to happen. If an event precedes an activity, then it is processed at the same simulated time of the activity start; if an event follows an activity, then it is processed at the same simulated time of the activity end.

As stated before, EADs have a hierarchical process structure. In fact, nodes are activities and events and even sub-processes. A sub-process is itself an EAD, therefore it can be zoomed revealing the included flow-chart. The EAD of a sub-process can refer to input and output ports of the including process, i.e. a sub-process is only a convenient arrangement for grouping an EAD sub-net and depicting it as a single node (this solution aims to improve model readability).

The different shapes for the main nodes of an EAD are depicted in Figure 1.13.

The core of the process behavior is designed adding directed edges. Edges are used *i*) for message passing and the evaluation of conditions through *functions* call, and *ii*) for the introduction of timing into process activity flows. We classified edges

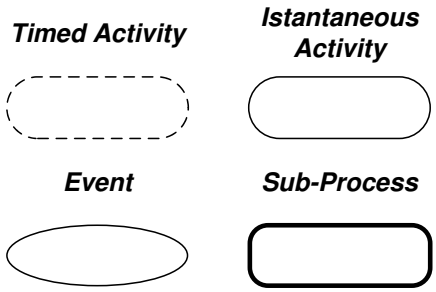


Figure 1.13: Nodes of an EAD.

in four types: *null edges* (or *true edges*), *inner edges*, *incoming edges*, *outgoing edges*. A null edge can be used only to connect events to activities or activities to activities. An inner edge connects an activity to another activity or to an event. These edges are depicted as arcs (see Figure **??**). Incoming edges are used to depict incoming messages from an input/output port, while outgoing edges depict outgoing messages to an input/output port. Incoming edges connect only activities to activities or activities to events, while outgoing edges also link events to activities. Edges may not have any node linked to the head, thus they work as a termination arc (i.e, the process becomes definitely terminated when one of these nodes is traversed).

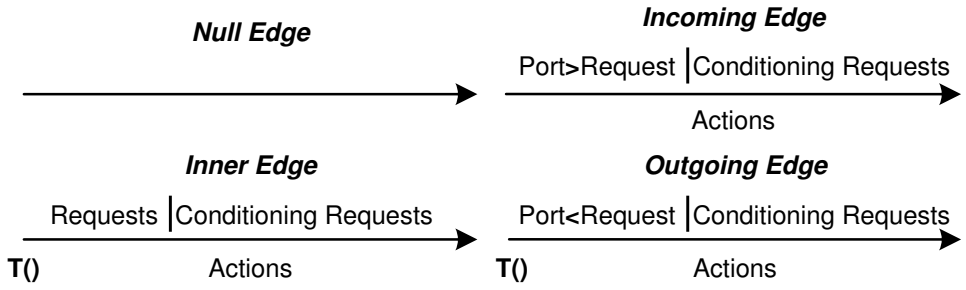


Figure 1.14: Edges of an EAD.

Edges produce a transition from the current enabled node (connected to the edge tail) to the future enabled node (that is linked to the edge head). To avoid deadlock, edge tail and head must be different nodes. While the transition rule for a null edge is always true, for the other edge types the edge traversal is allowed only if certain conditions are met. To understand the nature of these conditions, we introduce the

following edge functions: a *Timer* and a *Request*.

A Timer is a time function that generates a delay time using a probability distribution function (e.g., exponential) and the appropriate set of parameters. Only inner and outgoing edges may have a Timer. Whenever an inner/outgoing edge is selected by an activity to be eventually traversed, the Timer generates the simulated delay time and an "alarm" is scheduled to warn the edge after the delay time. When an edge has been warned, it is allowed to enable its head node (i.e., the process state has been changed), or rather it is traversed.

The transition rule of an edge is composed by a set of Requests. A Request is a (public or private) function of a process that can be called to check conditions, to set up model objects and entities parameters and (in case) exchange entities among model objects. If the conditions are met, the function returns true; false, otherwise. A set of primary Requests can be used on the same edge by the formulation of a logical expression using boolean operators (*and*, *or*, *xor*) and negation (*not*), that we call the *primary rule*; moreover, the verification of the primary rule may depend on a set of conditioning Requests, or what we call the *conditioning rule*. Both conditioning and primary rules compose the transition rule in the following way: "primary rule | conditioning rule". Only if the conditioning rule is evaluated as true, then the primary rule is evaluated.

All the edge types (excepted null edges) may have a transition rule. A considerable difference exists between inner and incoming/outgoing edges. In fact, by using incoming/outgoing edges, a process can: *i*) make a call to a public function of an external process (*receiver process*) linked to an input or output port; *ii*) receive a call to its own public functions by an outer process (*sender process*) linked to an input or output port. In this way we enable communication among processes (e.g., exchanging entities and assigning work). In particular, the primary rule of an incoming (outgoing) edge must only include the call of a Request by (of) another process linked to an input/output port. The symbolism used to call a Request of an outer object is "PortName<Request", while to depict the call of a Request by an outer object "PortName>Request" is used. For incoming/outgoing edges, the conditioning rule may also include calls to outer functions.

If a sender process calls a function of a receiver process, and this occurs only if at

least an incoming edge starting from the current node of the receiver has the called Request as primary rule, then the call to the Request function may return true; otherwise, false is automatically returned (i.e. the call is not accepted). Indeed, incoming edges work as triggers for processes that are waiting for an external input.

With the exception of null edges, which are always traversed, if an inner edge (or an outgoing edge) is selected to be traversed by the current node, then it is traversed once the traversal rule is true.

Timers and Requests are the condition functions that may be involved to cause an edge traversal. Nonetheless, inner and outgoing edges may have both a Timer and Request function. In this case, the execution priority states that the Timer must be executed before a Request, i.e. the transition rule is verified when the edge has been warned.

All the edges may also have a list of *Action* functions. An Action is a function that performs such operation, e.g. changing the parameters of the process or of an owned entity. If a Timer and/or a transition rule have been defined for the same edge, the Action must always be executed at last, i.e. after the edge has been warned and if true has been returned by the transition rule.

Each Request and Action can explicitly receive a set of parameters; therefore, the behavior of a Request/Action may change in function of the received parameters.

To improve MP scaling and model object reusability, as proposed in [78], each process may use *multiports*. A multiport is an indexed set of k ports named *portname[1],. . . , portname[k]*. A multiport is explicitly depicted as different ports in the external view of a model object; in the inner view, multiport may be used either in an explicit or implicit way by incoming/outgoing edges. For instance, an explicit use of a multiport may occur when the process needs to call a Request function of an external process linked to the port "*portname[i]*" (where the suffix *[i]* stands for the index of a port that is included in the multiport *portname[1,. . . ,k]*), then the name of the port is explicitly depicted in the transition rule of the function as follows: "*portname[i]<Request*". Another example is introduced to show the implicit use of a multiport. If the process needs to call the same Request function of the external processes linked to the multiport, the following notation must be used in the transition rule of the outgoing edge: "*portname[i:1,. . . ,k]<Request*". In this way, for

each *portname[i]*, where index i varies between 1 and k, the edge must check the associated transition rule and the edge is traversed whenever at least one transition rule is true. The implicit use of a multiport allows our MP to be compact and to easily implement the behavior of dynamic processes. Moreover, by using implicitly a multiport, it is possible to specify a sub-set of port indexes that *i*) must be called through an outgoing edge or *ii*) allow an external process to call an inner Request through an incoming edge. The sub-set of port indexes can also be a list name generated at runtime using an Action function.

Our MP allows defining a process behavior via EAD using activities, events, sub-processes and directed edges. Another type of nodes, called *logical nodes*, has been defined in order to support the definition of process paths. These nodes are: *i*) *boolean fork*; *ii*) *split*; *iii*) *and*; *iv*) *or*; *v*) *unconditional flow* (Figure 1.15). A logical node may be used to represent alternative paths to be chosen under specified conditions.

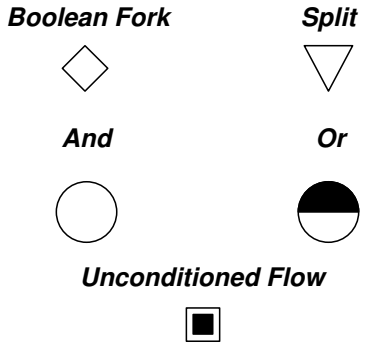


Figure 1.15: Logical nodes of an EAD.

A boolean fork can be connected to the head of a conditional edge, or rather those edges that have a transition rule. Using this node, if the transition rule is true, then the edge is traversed and the process control flows through an edge starting by the boolean fork node; otherwise, the edge is also traversed, but the process control flows through a special edge, depicted with a dashed line, which starts by the boolean fork node. Therefore, using the boolean fork node after a conditional edge, the edge is always traversed. The outgoing edges from a boolean fork node are of the null or inner type. In case inner edges are used, another boolean fork must

be used for each inner edge to catch the result of the transition rule.

The unconditional flow node acts in a similar way of a boolean fork, but whatever be (true or false) the transition rule of the incoming edge, only one edge must leave this node.

The remaining logical nodes may be connected to the head of any edge type. Once the edge is traversed, the process control passes at the logical node. A split node is used to separate the process path in two or more alternative paths. When the process path is separated in more paths, to recombine two or more paths, an and/or node is required. The and node, becomes enabled at the time all the edges incoming to the node have been traversed. The or node become enabled at the time at least one of the edges incoming to the node has been traversed.

The and, or and split nodes can be part of the process state. Typically the process state has only one active activity per process state. However, once the process flow is separated by using a split, many activities and the split/and/or nodes may be enabled, thus they may be part of the process state.

Some explanations are required about the use of sub-processes as nodes of an EAD. As they are EAD nodes, edges that start and arrive to these nodes are of the types defined in the previous. The only restriction is for edges that start by a sub-process node. If an edge that starts from a sub-process node is not a null edge type, a boolean fork must catch the edge transition results (otherwise the process checkpoint may stay on an edge included in the sub-process). The inner view of a sub-process is a particular EAD which consists of at the most one edge that links the outer view to an inner node (or rather from the inner boundary to a node) and/or at the most one edge that links an inner node to the outer view (or rather from a node to the inner boundary). Also if the edge that starts from the sub-process inner boundary to a sub-process node is not of null edge type, a boolean fork must catch the edge transition results (otherwise the process checkpoint may stay on an edge included in the sub-process). An example about a finite queue is shown in Figure 1.16.

Processes are usually executed during more simulation time periods. For this reason the nodes of a process can be visited during different time periods. To track this possibility, we adopt the *process checkpoint*. A process checkpoint is a property

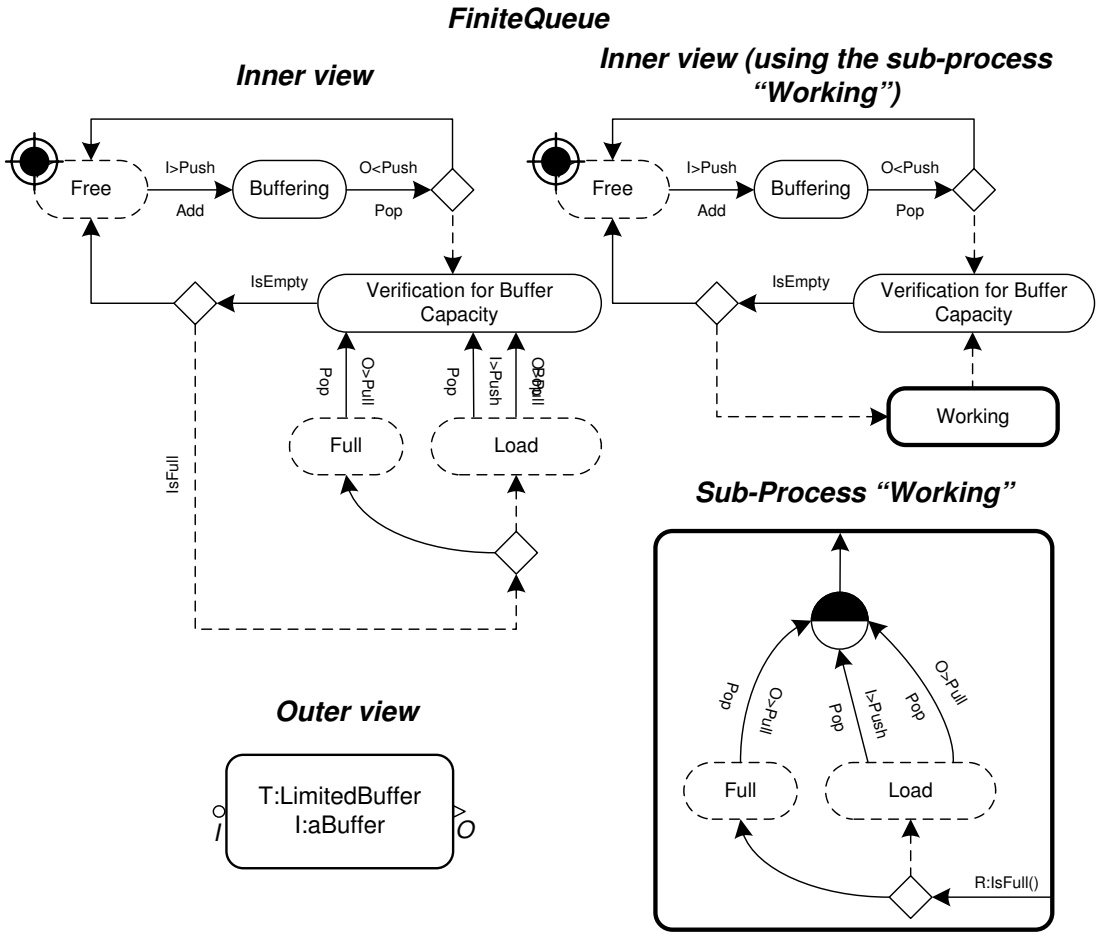


Figure 1.16: An example of use of a sub-process for a model object that is designed as a buffer with a finite capacity.

of the processes that shows the set of nodes from which a process starts or is reactivated, i.e. the checkpoint locates the enabled nodes (the process state). A process checkpoint is also called *reactivation point*. For each process, the reactivation point must be explicitly shown to depict the initial active states as done in Figure 1.16 for the "Free" node (that is the initial state of the FiniteQueue model object): the reactivation point is depicted as a "target" and is usually placed on the upper-left corner of a node.

In HMP simulation models, a process can have the following status in the sense

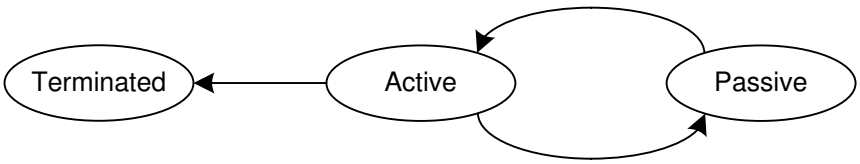


Figure 1.17: EAD process simulation status.

of simulation execution: *active*, *passive* or *terminated* (see Figure 1.17). Focusing on a non-concurrent simulation, only one process can be active at a time. A process is active in the sense that it is moving through its paths, until it enters the passive state or is terminated. A process enters the passive state when a Timer schedules a delay time or a message is sent to an input/output port. In a similar way, a process becomes active by means of an external trigger (a message from an input/output port) or a previously scheduled warning time. To start model simulation, at least one process must receive a message to be activated for the first time. An active process may become terminated whenever an outgoing or transfer edge is traversed and the edge head is not linked to any node. A terminated process returns false to each incoming message and will never be active during the simulation.

1.6 Modeling of a Whole Maritime Container Terminal

As stated into Section 1.1, freight transportation companies are focused on the study and development of more efficient and effective transport systems.

In this context, maritime container terminals cover a key role in the freight transportation network. The common, basic idea in freight transportation is that of resorting to the aggregation of multicommodity flows to be transported, by large carriers, for long distance routes across a certain number of intermediate logistic platforms (terminals), where freight consolidation may occur together with store and forward logistic functions and other source/sink based functions. Standardized containerization appears as the most notable and consolidated practice for freight

transportation, especially through long distance maritime routes. Containerization is a system of intermodal freight transport using standardized large bins known as containers that can be loaded and sealed intact using multiple modes of transportation: ship, rail, road and air. Containers are an ISO standardized metal box of 8-ft wide by 8-ft high; the most common lengths are 20-ft and 40-ft. The container capacity is often expressed in twenty-foot equivalent unit (TEU). Containers are made out of steel and can be stacked on top of each other.

Nowadays, approximately 90 percent of the non-bulk cargo worldwide moves by containers stacked on transport. On ships they are typically stacked up to seven units high and there are ships that can carry over 9,000 TEUs. The world container fleet amounts to about 23.2 million TEUs and the container throughput reached 440 million TEUs in 2006 [90].

Maritime container terminals are the most important crossroads for transshipment and intermodal container transfers. The maritime container shipment follows the spokes-hub distribution paradigm: containers are shipped from a port (spoke) to another one through a small number of maritime transshipment terminals (hubs). Both spokes and hubs could be connected with inland container terminals by road and rail (the so called intermodal transfer). In opposition with the high number of container ports in the world, there are a few of transshipment terminals (e.g., the *Gioia Tauro terminal*). Transshipment container terminals are large facilities for intermodal transport, able to handle millions of containers per year and berth large container vessels. These hubs are linked with spokes by means of small vessels called *feeders*, through minor (or *short sea trade*) routes. There are a few major liner trade routes that link the hubs (or *deep sea trade routes*). There are also oceanic container vessels, known as *mother vessels*, which sail on these intercontinental routes.

A maritime container terminal is a large and complex logistic platform organized around a set of logistic processes. The logistic activities at a container terminal often belong to more logistic processes, which require a whole vision of the system to be properly organized. This fact is critical for a good management of the system and the choice of the system modeling approach: an efficient and effective management of logistic activities in a container terminal can decrease the operating costs and service times and increase the quality of services in order to achieve a better market

position.

Thus, in this Section we report the high level schema of a simulation model of a whole maritime container terminal. We originally proposed the model in in [42]. The availability of a simulation environment where decision models could be developed and tuned according to a final customization phase is considered as a novel, challenging market option.

In the following, we briefly classify and describe the logistic problem at hand. Afterward, we depict the queuing network model used to represent the core logistic processes at a container terminal. The model has been conceived using a holistic approach and the model can be developed using some of the MPs described above.

Problem classification Vis and De Koster [91] produced an interesting overview paper in the area of container logistics that gives a classification of the logistic processes in modern container terminals: *i*) *arrival of the ship*, *ii*) *discharging and loading of the ship*, *iii*) *transport of containers from ship to stack (and vice versa)*, *iv*) *stacking of containers*, and *v*) *inter-terminal transport and other modes of transportation*. With respect to this classification, it is possible to identify a set of features that are common to many maritime terminals. The main difference between the greater part of container terminals located in Europe and North America and those in the Asia-Pacific region regards the logistic processes *ii*) and *iv*). The latter relies on the "Indirect Transfer System" (ITS), in which process *iii*) and *iv*) are closely connected: a fleet of shuttle vehicles transports the containers from a vessel to beside the stack area (and vice versa) while dedicated cranes stack containers in compact regions. European and North-American container terminals are generally based upon the "Direct Transfer System" (DTS) in which process *iii*) and *iv*) are performed by the same actors: in this case a fleet of SV called SCs moves the containers from a vessel to the storage area (and vice versa) for container stacking (retrieval) operations into (from) the slots assigned in the storage area [15]. In this study we refer to a DTS maritime container terminal.

1.6.1 A Simulation Model for a Marine Container Terminal

To fix ideas, we give a brief description of a real case that we are familiar with: the *Gioia Tauro terminal*.

This terminal, located in southern Italy, since it opened in 1995 has become in just a few years one of the largest transhipment port in Europe. Its management has been early characterized by significant efforts towards an increasing computer aided organization and control of the various logistic processes. Recently a manager friendly simulator has been designed for studying the congestion phenomenon at the port-input channel, the port admission policy for the newly arrived vessels in the roadstead and the allocation policies of berthing slots [11]. Now, the problem of achieving an integrated management based on the usage of simulators and other operations research tools becomes more and more important.

Description of the container terminal The terminal is a large facility composed by: a harbor entrance of 250 m wide and 18 m water depth followed by a large roadstead for incoming vessels; 2 pilot boats; a 3,100 m quay length with along a channel of multiple water depth ranging from 13.5 to 15.5 m (the quay is discontinuous, thus it is usually decomposed into two sub-quays); 18 quay *rail-mounted gantry cranes* (RMGCs) and 5 *rubber-tired gantry cranes* (RTGCs); a fleet of 75 SCs, handling vehicles used to transfer the containers between the quay cranes and the yard; and a yard surface of 1.1 million square meters that can store nearly 59,000 TEUs. The yard has 32 sectors parallel to the quay and organized in two lines. An average quay has 32 lines, each containing 16 slots. A slot can host two one-TEU containers stacked one on top of the other. The layout of the container terminal is depicted in Figure 1.18.

Description of the Logistic Processes The logistic processes are described in the following.

Planners of the terminal company construct a "berth schedule" on a weekly basis, which shows the berthing time and position for each incoming vessel according to the *ETA* (Expected Time of Arrival) and *DTD* (Due Time of Departure) of the

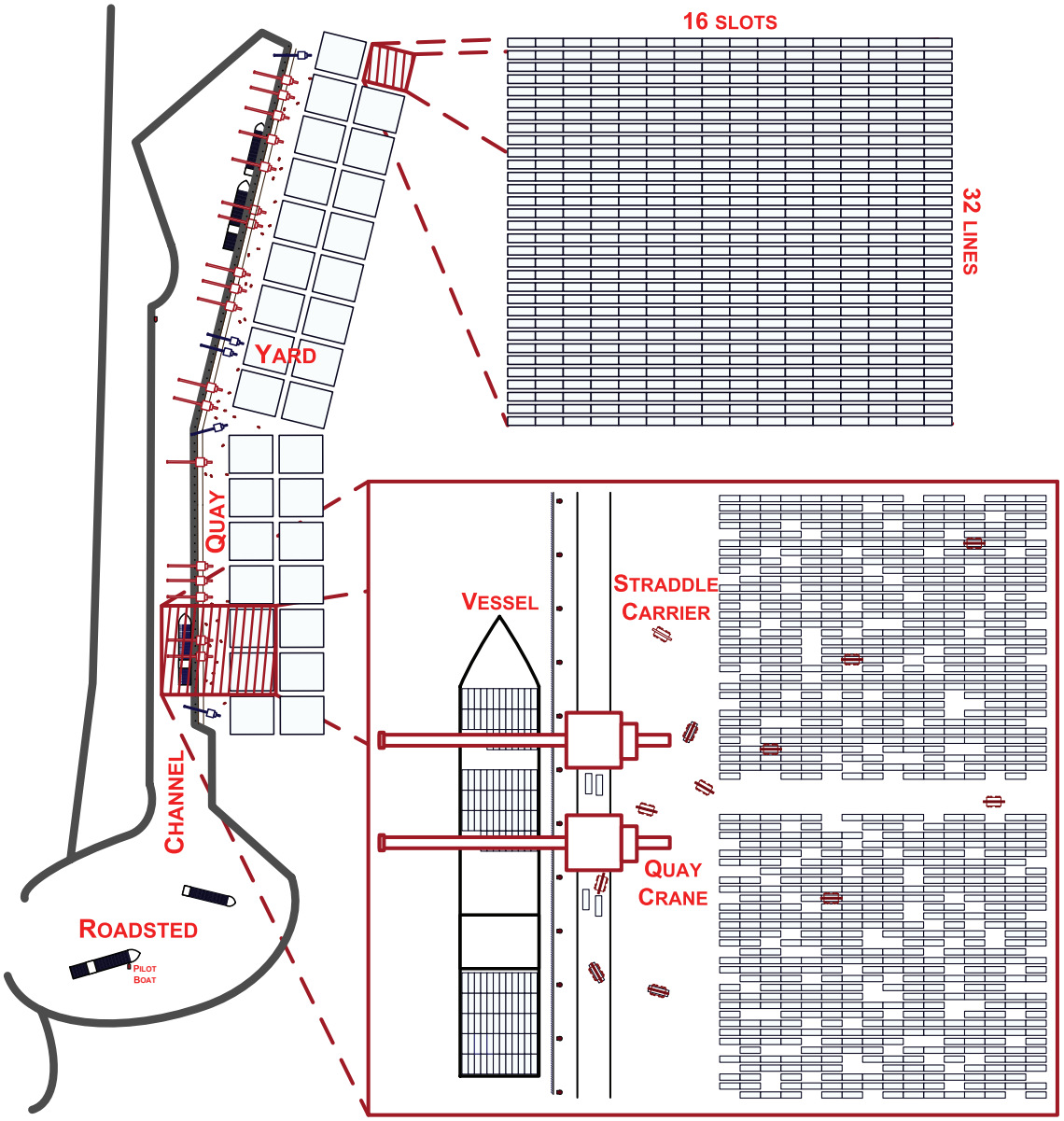


Figure 1.18: The layout of the Gioa Tauro Maritime Terminal. Special zoom on a yard sector and on the loading/unloading and containers transport processes.

vessel and the preferred sub-quay. This is the so called "Berth Planning Problem" [60, 16]. The goal is to find the optimal berth position for each vessel, i.e., the berth position that minimizes container handling cost from the vessel to location

in the marshalling yard where outbound containers for the corresponding vessel are stacked. In Figure 1.19 there is an example of a berth schedule. We assume that the berth schedule is an exogenous input to the part of the simulation model discussed here. Actually, vessel entrance at port and berth schedule may be affected by the specific features and physical characteristics of the real port of interest, but, here we focus on common logistics connected with other basic operations to be performed on berthed (standard) vessels by (standard) QCs.

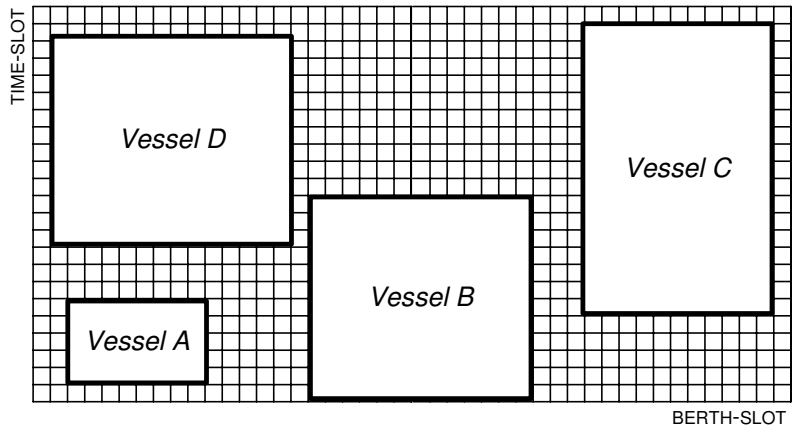


Figure 1.19: An example of berth schedule.

The arrival of the ship process proposed in [43] is the starting point for a new generalized model and is described in the following. Once the vessel has arrived outside the port (the arrival time is an uncertain time close to the vessel ETA), its mooring along the assigned berth position depends on the following requirements: *i*) *formal conditions* (e.g. contractual agreements between the vessel's shipping line and the port of call for the use of port facilities), which means a priority policy for the port entrance queue; *ii*) *operational settings* (i.e. pilot mariner and pilot boat availability, berth space assignment and the least number of free quay cranes). If requirements are met, the ship is manoeuvred down the navigation channel and into its berth slots by one or two pilot boats; otherwise it must wait in the roadstead. In the model the ship population is finite and corresponds to a specific set of feeders and mother vessels.

Once a vessel is at berth, container discharge/loading can be initiated only if

mechanical (and human) resources are allocated; if not, the ship waits in its berth position until resource assignment. Discharge/loading operations are performed by QCs placed along the berth: one or multiple cranes move containers between the ship and the quay area according with a pre-established operation plan. The maximum number of QCs that is possible to assign to each vessel is restricted by *i*) the total number of cranes in the quay and *ii*) the maximum number of allowable cranes to each vessel due to physical (i.e. the length of the vessel) and logical constraints (i.e. interference between cranes). Considering the span of the cranes (about 30 m) and the horizontal space necessary to stack and transfer away the incoming/outgoing containers of a vessel, the maximum number of cranes assignable to the longest vessel is 5 (this number is proportionally decreased for shortest vessel). The problem of assigning QCs to the vessels for each time-slot is called "Quay Crane Deployment Problem" (QCDP) [41, 39]. This problem is crucial for the design of an intelligent resource allocation. In Appendix A we propose a preliminary study on the QCSP.

When multiple cranes are assigned to the same ship, crane interference has to be avoided and a complex scheduling problem arises to manage the relationships (precedence and mutual exclusion) existing among the deck/hold tasks of the same ship. This is the "Quay Crane Scheduling Problem" (QCSP) [12] (the QCSP is deeply discussed in Chapter 3).

Both problems can be managed together in a two-phase approach: the first phase concerns the QCDP, while the second one involves the QCSP. Assuming that the berth schedule is deterministic for the model, in the first phase, using such heuristics or commercial IP solver, an assignment for every vessel at each one-hour time-slot a sub-set of QCs is found [39]. Afterwards, in the second phase we use a metaheuristics to schedule the ship's tasks to the assigned QCs [44]. In this way, the simulation model is initialized.

The performance of the discharge/loading process highly depends on the availability of this type of cranes and their turnover speed. Again, this problem has been deeply studied within the real context of Gioia Tauro Terminal [12], but without being integrated with the problem of simulating yard organization, SCs travel back and forth from the yard and analyzing the effects of crane shifting along the berth, under the condition that multiple vessels are at berth.

In a DTS container terminal the transport of containers from ship to yard (and vice versa) and the container stacking processes are performed by a fleet of straddle carriers (SCs). SCs take in charge containers and cycle between the berth area and the assigned storage positions within the yard. Straddle carriers are capable with a laden container of relatively low speeds (up to 20–26 km/h). The yard is a passive resource: it consists of a matrix of 2 lines and 16 columns (this is an average value), in which each matrix element is a three-dimensional matrix that stands for a yard sector. SCs are able to select a slot within the yard structure to load/unload or stack containers. They are also able to compute the distance from the assigned QC to the yard slot and back.

Each QC has about four SCs assigned. An optimization code to dynamically assign SCs to QCs may be useful.

In the present case, no considerable inter-modal TEU transportation will be considered.

A Queuing Network Model Description The model has been designed using a holistic approach. As explained in Section 1.1 at page 1.1, the model represents the interaction between different logistic processes through special model objects called resource managers. Resource managers are gifted of a high-level view of the whole system and are able to operate on the system resources. They use exogenous data (i.e. berth schedule, QC deployment and scheduling) to dynamically assign the container terminal resources to the jobs.

The logistic processes described above have been depicted using queuing networks. The arrival of the ship process is described in Figure 1.20. The model is based on the hierarchical representation proposed by Legato and Mazza [43]. Incoming vessels wait in a priority queue that represents the roadstead; outgoing vessels wait in another priority queue that stands for served vessels waiting in the assigned berth slot. A resource manager called "Berth Manager" assigns pilot boats to incoming and outgoing vessels according to previously defined conditions (i.e., active and passive resource availability).

A vessel entering the rightmost (black) box in the figure means that it has been berthed and another queuing network model takes charge of it.

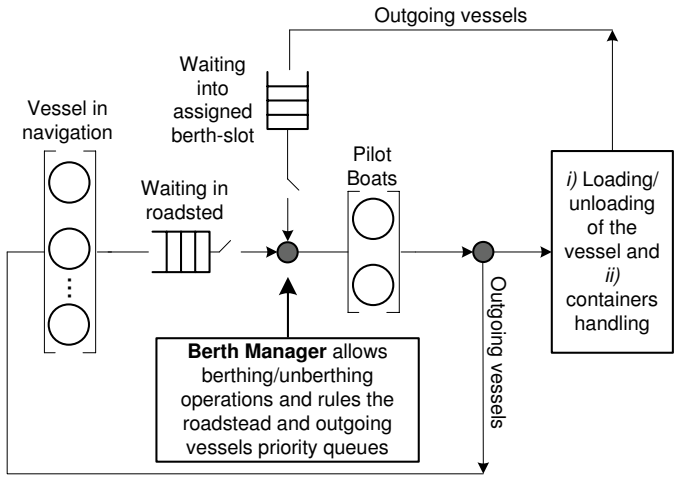


Figure 1.20: Arrival of the ship process

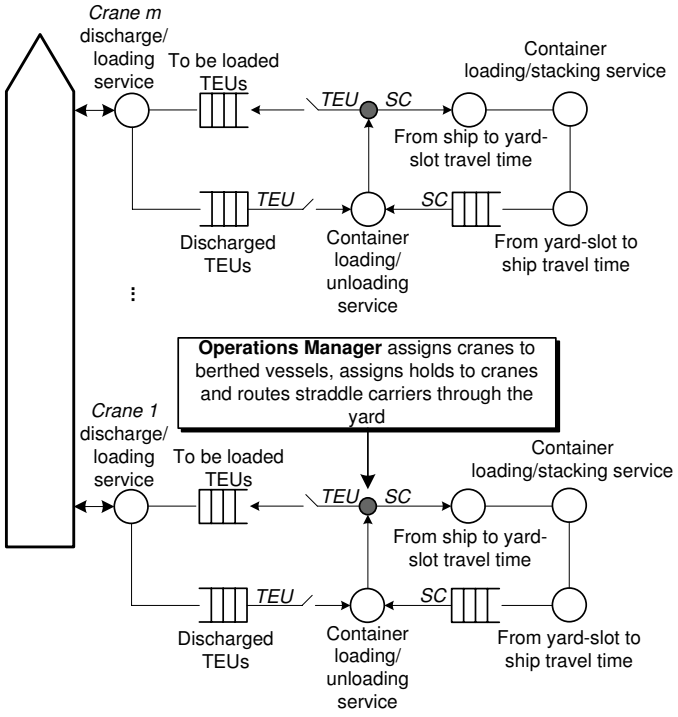


Figure 1.21: Discharging/loading of the ship, containers transfer and storage/retrieving processes.

Figure 1.21 shows the queue network model for the loading/unloading of the ship process and container transport from the ship to the assigned yard-slot (and vice versa) and the consequent storage process. The simulation model uses a list of berthed vessels. Each vessel is assigned to a berth slot (a discrete part of the quay assigned in the berth schedule). For each vessel in list, a resource manager called "Operations Manager" assigns cranes to the vessels and starts the operations on the vessels. When discharging and/or loading operations have been performed, the Operations Manager pushes the served vessel out of the model and updates the system performance measures (e.g., the vessel's overall completion time). In Figure 1.21, both the discharging and loading process of the ship are depicted. The Operations Manager tasks are: *i*) for each vessel in the berth, to assign the vessel0s tasks to be served to each QC; and *ii*) to route SCs within the network in order to execute the optimal handling operations.

In the following, we concentrate our attention on the model shown in Figure 1.21 and in particular on the discharging operations of a particular vessel. We aim to show how the same model can be developed using different modeling approaches.

1.6.2 A Modeling Case Study

In this section, a real case study of modeling of a complex logistic process is presented, with the aim of comparing the expressiveness of our MP with other well-known approaches.

The system that we are intended to model concerns the management of quay crane operations on a specific vessel. We present a special case of the model depicted in Figure 1.21: the discharging and the related container transfer from ship to yard process is described. Details on container storage are not provided. Moreover, container retrieving and transfer from the yard to the quayside and vessel loading are not modeled.

To fix ideas, we give a more detailed description of the system that we are modeling, with respect to the description provided in SectionContainerTerminalSection. Once a vessel is moored, a known number of containers must be discharged from the vessel's bays (i.e. from the deck or the hold of a bay) according to the operation

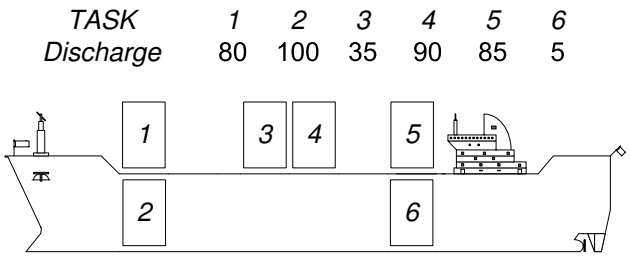


Figure 1.22: Map with discharge info per vessel task.

Crane	*Task Sequence*
Crane 1	Task 1, Task 2, Task 3
Crane 2	Task 5, Task 6, Task 4

Table 1.1: Discharging sequence.

plan shown in Figure 1.22.

As described in Section 1.6, discharge operations are performed by some QCs placed along the berth, which move containers between the ship and the quay area. When multiple cranes are assigned to the same ship, crane interference has to be avoided and a the QCSP must be resolved to shun the violation of precedence and non-simultaneity constraints existing among vessel's tasks.

Here, we are interested in depicting the container discharge process of a certain vessel and the container transfer from the limited buffer area (max 6 containers) under each QC to the yard side by means of SCs (as described in Section 1.6) .

Since the process of transferring containers from the quayside to the yard is performed by SCs that are used to cycle among these areas with (outward path) and without containers (backward path), this process may also be depicted as a unique service time, or rather "the yard cycle time", which includes the outward time, the container unloading time and the backward time.

In the following, we suppose for simplicity that two QCs must simultaneously perform the discharging operations of the vessel depicted in Figure 1.22. In Table 1.1 is reported the sequence of tasks that must be discharged by the two cranes.

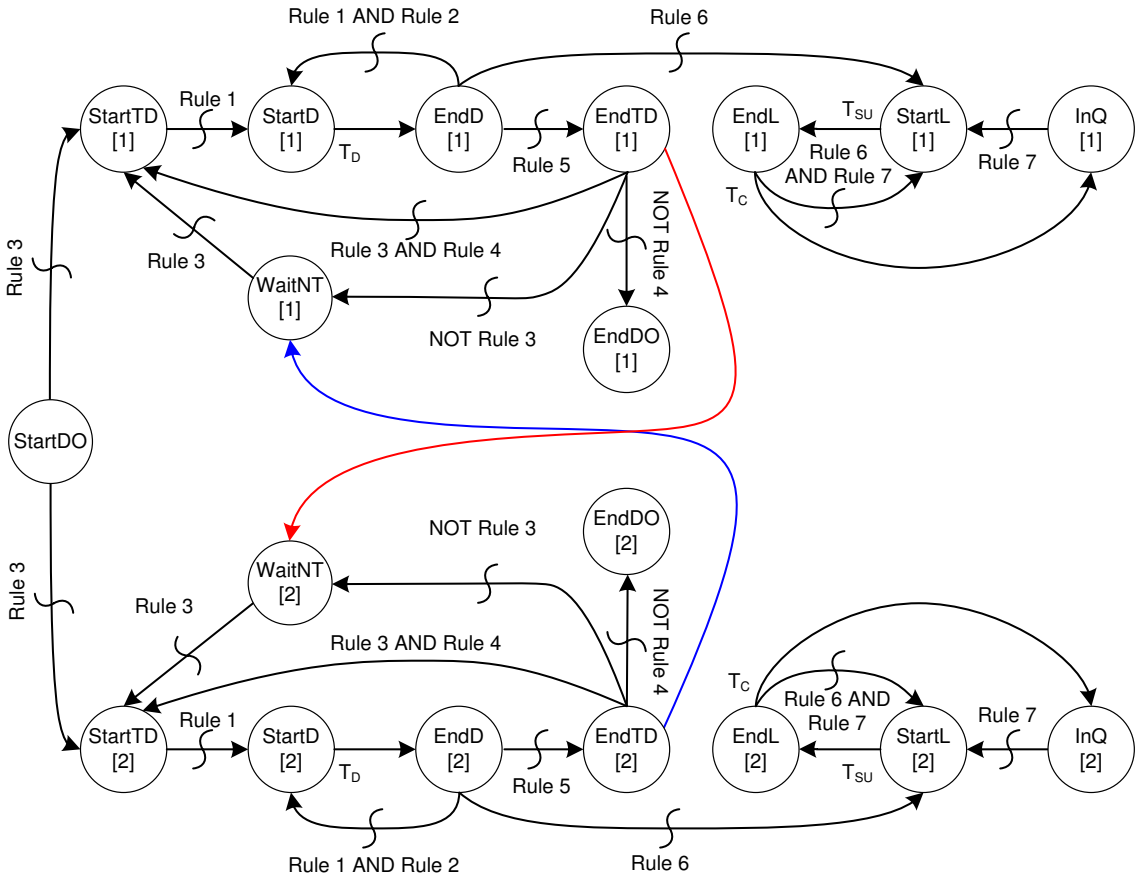


Figure 1.23: The event graph of event graph of the quay crane operations model.

Event	*Description*	*State Changes*
StartDO	Cranes start discharging operations	For each crane *i*, set *current task* using the index of the first task to be discharged in the crane i-*th* discharging sequence
EndDO[i]	Crane i-*th* ends assigned discharging operations	Set *current task* to *null*
StartTD[i]	Crane i-*th* starts discharging of *current task*	Set *current task* status to *in progress*
EndTD[i]	Crane i-*th* ends discharging of *current task*	Set *current task* status to *free*

Event	*Description*	*State Changes*
StartD[i]	Crane i-*th* starts discharge of a container from the current task	
EndD[i]	Crane i-*th* ends discharge of a container from the current task	Decrease by one the "number of containers" in *current task*. Improve by one the size of "buffer area under crane i-*th*"
WaitNT[i]	Crane i-*th* waiting to acquire the next task	Set *current task* using the index of the next task to be discharged in the crane i-*th* discharging sequence
InQ[i]	Straddle carrier arrival in quay	Improve by one the size of "waiting line in Quay to crane i-*th*"
StartL[i]	SC starts container loading	Decrease by one the size of "waiting line in Quay to crane i-*th*"
EndL[i]	SC completes container loading	Decrease by one the size of "buffer area under crane i-*th*"

Table 1.2: Events of event graph of the quay crane operations model.

Edge Condition	*Description*
Rule 1	Buffer area under crane capacity is less than 6
Rule 2	Hold has not been completely discharged
Rule 3	No violation of precedence and non-simultaneity constraints
Rule 4	Crane has more tasks to be discharged
Rule 5	Current task is completely discharged
Rule 6	At least one SC waiting in quay
Rule 7	At least one container under the crane

Table 1.3: Enabling conditions of event graph of the quay crane operations model.

Figure 1.23 shows the Event Graph model of the logistic processes described above; events are labeled with numbers in squared brackets that refer to a specific crane.

Table 1.2 and 1.3 report the descriptions of the events (the vertices of the graph) with the related state change performed once an event is selected to be executed, and the edge conditions. The edge delay time functions T_D, T_{SU}, T_C, are respectively: *i*) container discharging time for each QC, *ii*) container set-up time for a generic SC, and *iii*) cycle-time for an SC (i.e. the time to transfer a container from the quay to the assigned yard-slot, to set-down the container and, finally, to move back to the assigned QC). As a matter of fact, the possibility to discharge a task by a crane is due to the respect of the precedence and non-simultaneity constraints. In this way, the availability of a task depends on the current task that is assigned to the next crane. In our example, if *Crane 1* is discharging *Task 3*, then *Crane 2* must wait the completion of *Task 1* to discharge *Task 3* (and vice versa). To depict this process, every time a crane completes its discharging operation on a task, then it warns the next crane (edges in blue and red).

Using Petri Nets, the system may be modeled as depicted in Figure 1.24. In this model, seeing that the number of tokens in some places were too large to be explicitly indicated (e.g. the tokens which stand for the containers to be discharged) we have directly put in each place the number of tokens. Using Petri Nets, we are able to design the non-simultaneity constraint between *Task 3* and *Task 6* (edges and nodes in blue), as well as precedence between tasks to be discharged (inhibitor arcs in red). Nevertheless, this approach does not provide reasonable model readability, and the net explosion is inevitable as the number of tasks and constraints rise up, as in real cases.

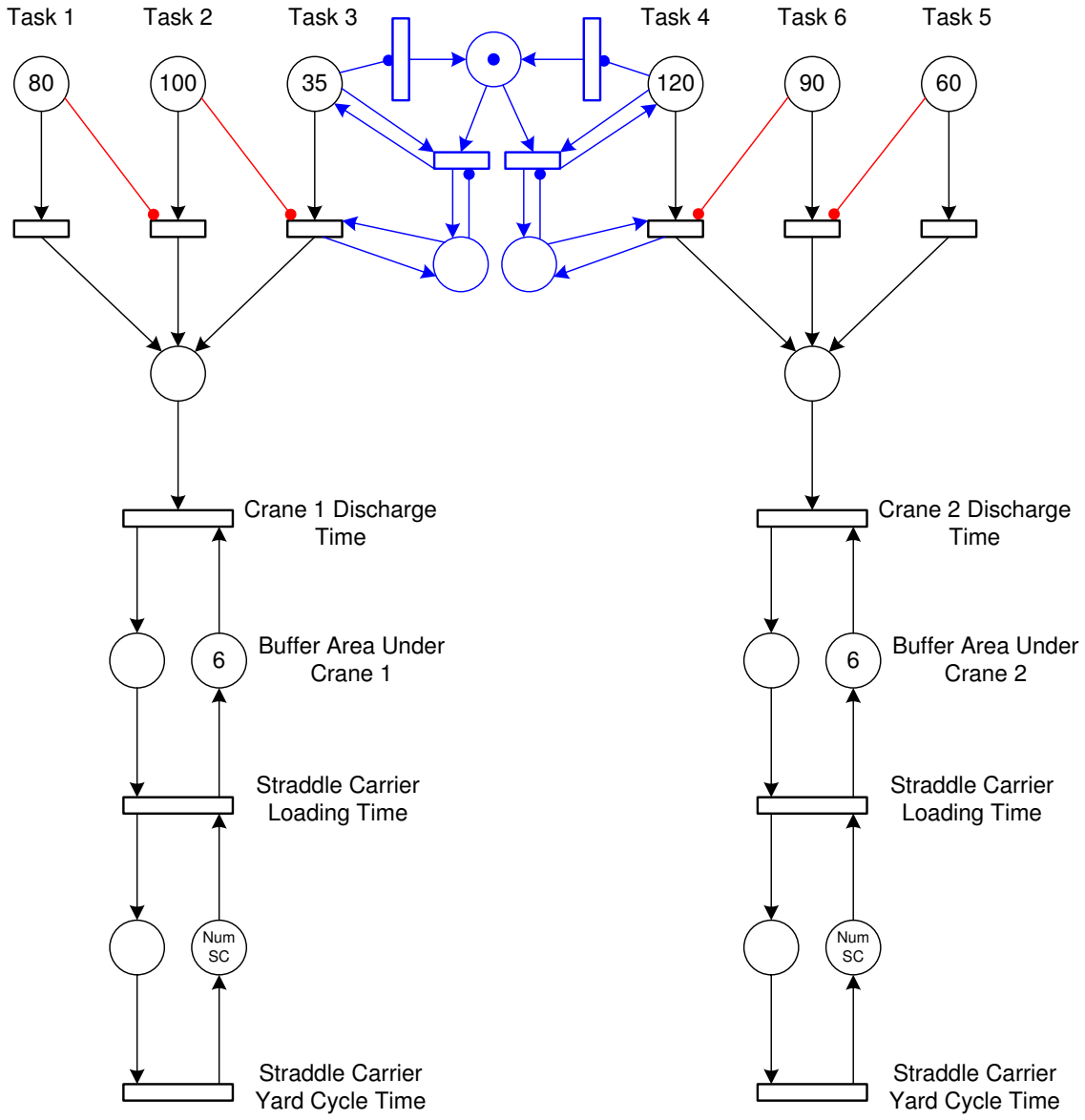


Figure 1.24: The Petri Net for the quay crane operations model.

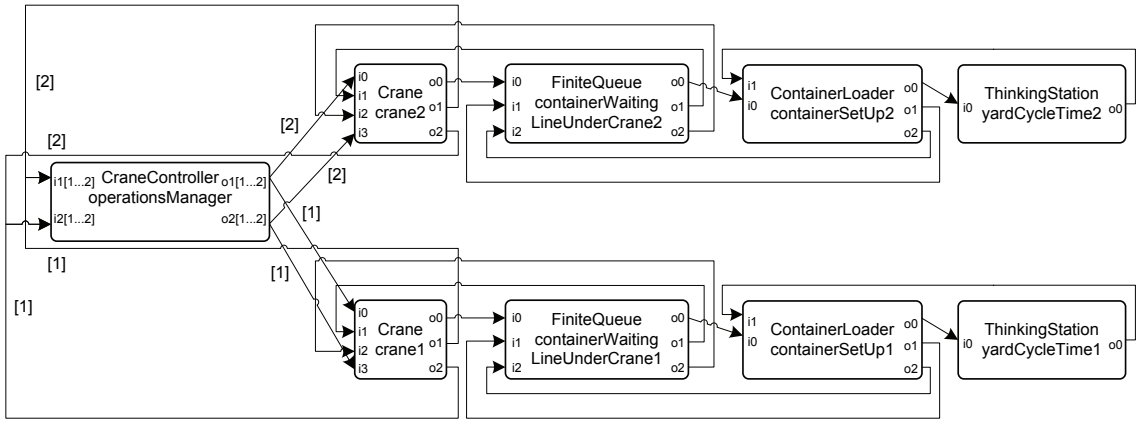
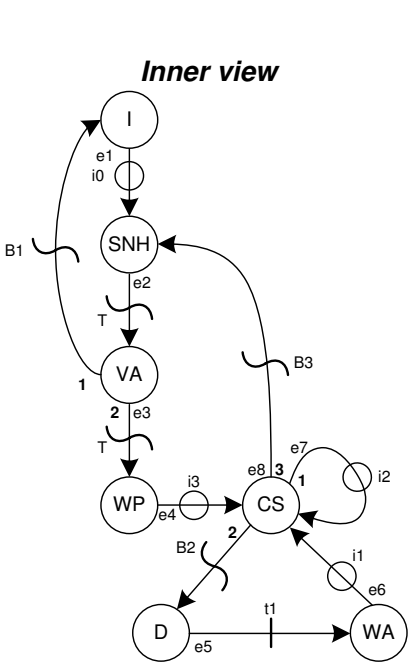


Figure 1.25: The HCFG Model for the Quay Crane Operations Model.



Outer view

i0 (scheduleInput) i1 (ackInput) i2 (pushOut) i3 (useNotification)	Crane istanceName	o0 (ContainerOutput) o1 (holdUseRequest) o2 (holdCompleted)

Local Variable(s)
type Task : nextTask
type Distribution : DischargingTimeDistribution
type Boolean : locked

Event(s)

e1:
Receive message TASKLIST from i0;

e2:
If TASKLIST is empty then
Set nextHold = null;
Else
Remove first hold from TASKLIST;
Set nextTask = first task;

e3:
Create message USE_PERMISSION;
Set USE_PERMISSION = nextTask;
Send message USE_PERMISSION to o1;

e4:
Receive message USE_PERMISSION;
locked = false;

e5:
Decrease by one the number of container in nextTask;
Create message Container;
Send message Container to o0;

e6:
Receive message ACK from i1;
If ACK = TRUE then
locked = false;
Else
locked = true;

e7:
Receive message PUSH_OUT_Container from i2;
Create message Container;
Send message Container to o0;
Set locked = false

e8:
Create message TASK_COMPLETED;
Set TASK_COMPLETED = nextTask;
Send message TASK_COMPLETED to o2;

Boolean predicate(s)

B1: nextTask = null

B2: number of container in nextTask is greated than zero and locked is not true

B3: number of container in nextTask is zero and locked is not true

Time-delay function(s)

t1:
Generate discharging time using DischargingTimeDistribution;

Figure 1.26: The HCFG and the corresponding external view of the Crane AC.

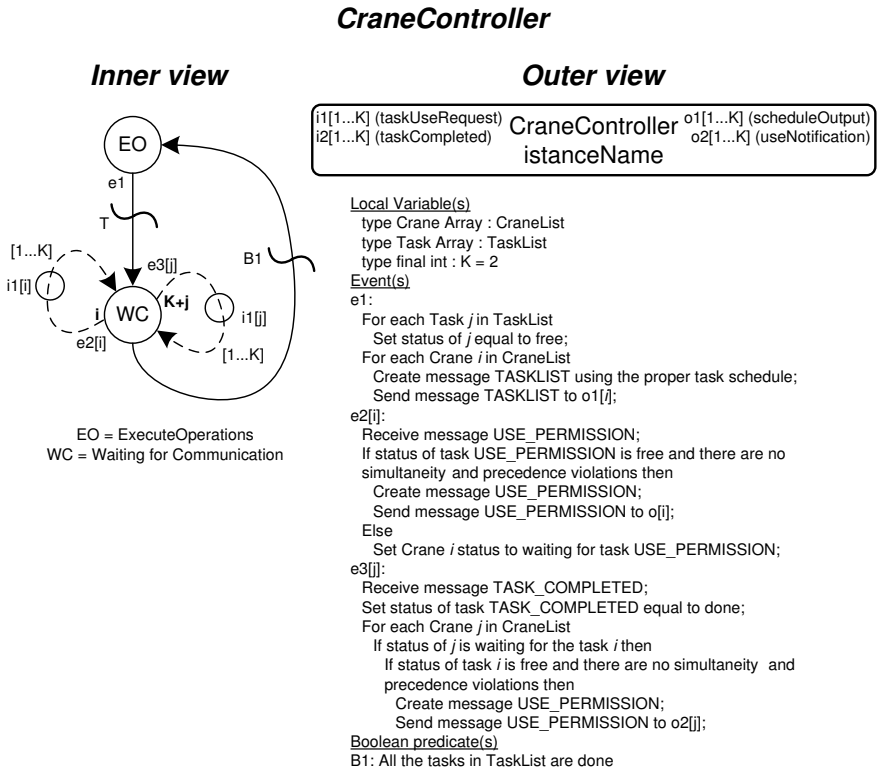


Figure 1.27: The HCFG and the corresponding external view of the CraneController AC.

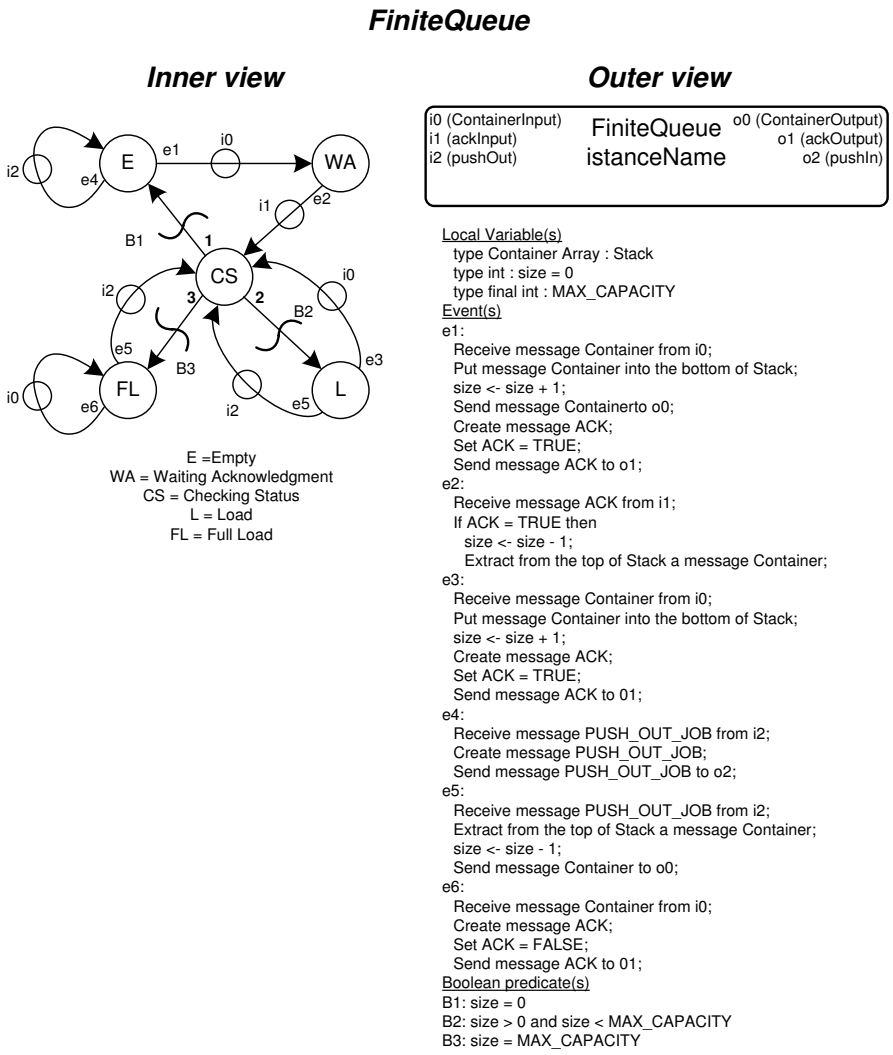


Figure 1.28: The HCFG and the corresponding external view of the FiniteQueue AC.

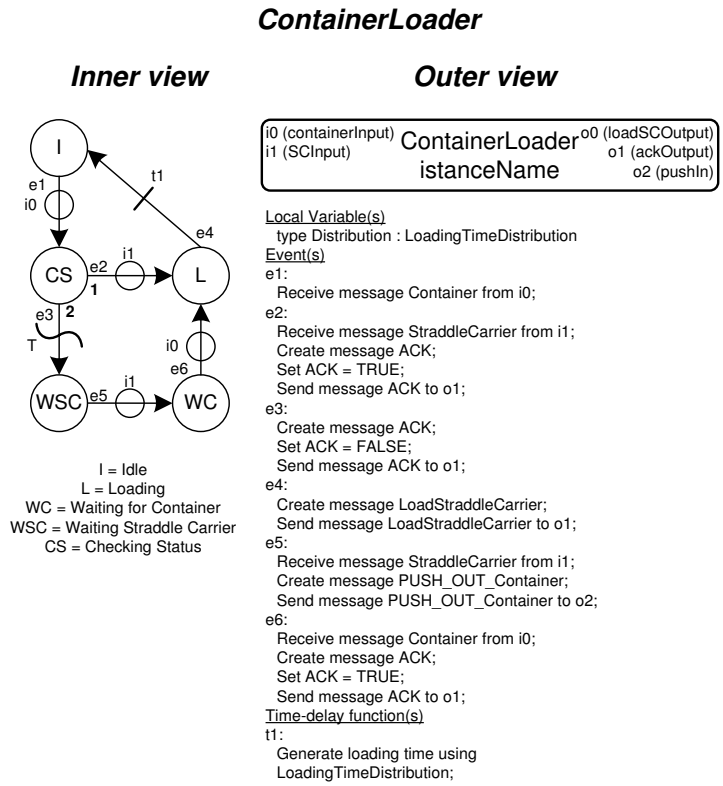


Figure 1.29: The HCFG and the corresponding external view of the ContainerLoader AC.

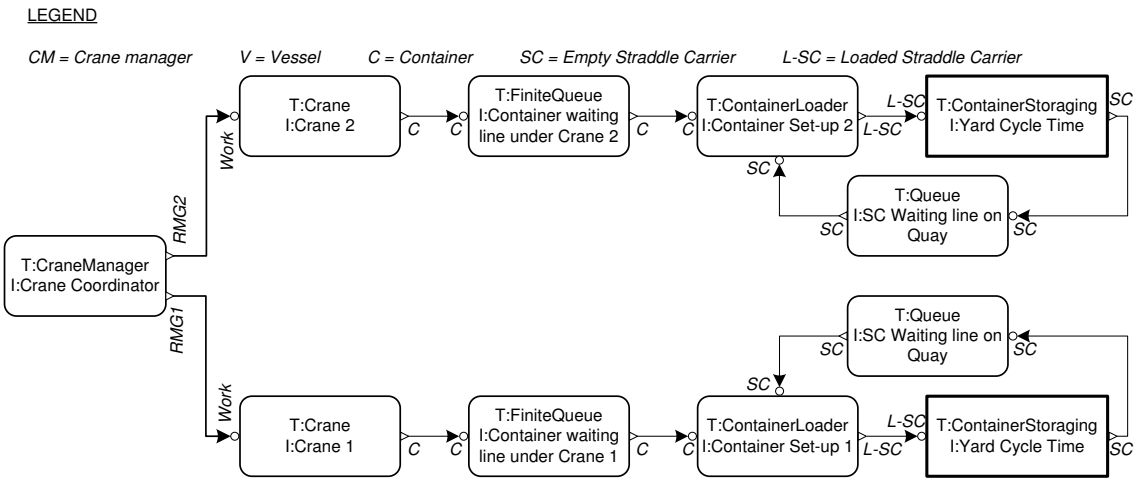


Figure 1.30: The HMP for the quay crane operations model.

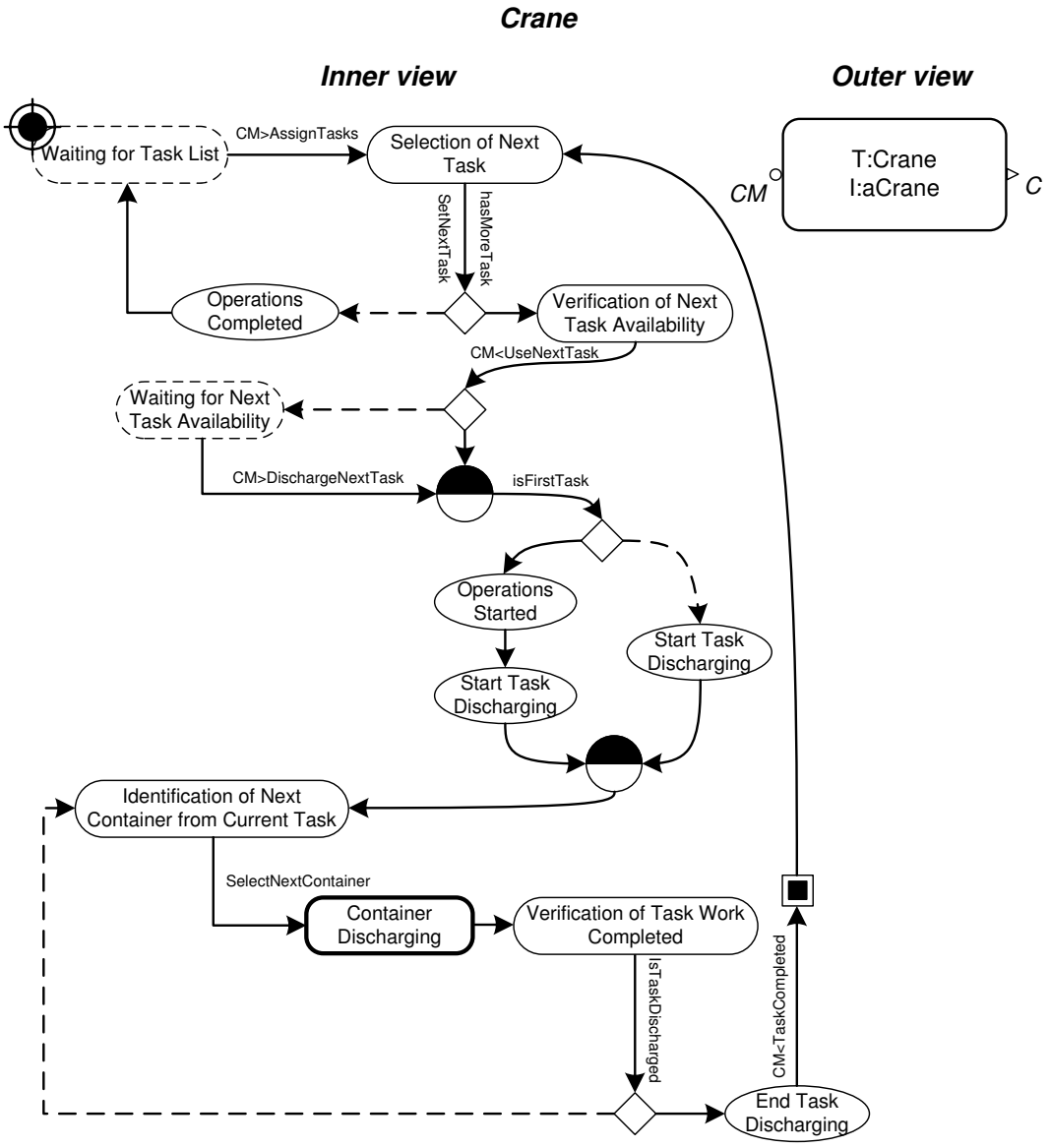


Figure 1.31: The EAD and the corresponding outer view of the Crane model object.

In the following we present the HCFG model for the discrete system at hand. Sargent [78] proposed a useful and wide set of ACs and CCs for the modeling of queuing systems. Unfortunately, as underlined by Sargent in its work, more effort is required to develop HCFG models of complex communication networks or manufacturing systems. The reason can be lead back to the need of the use of finite

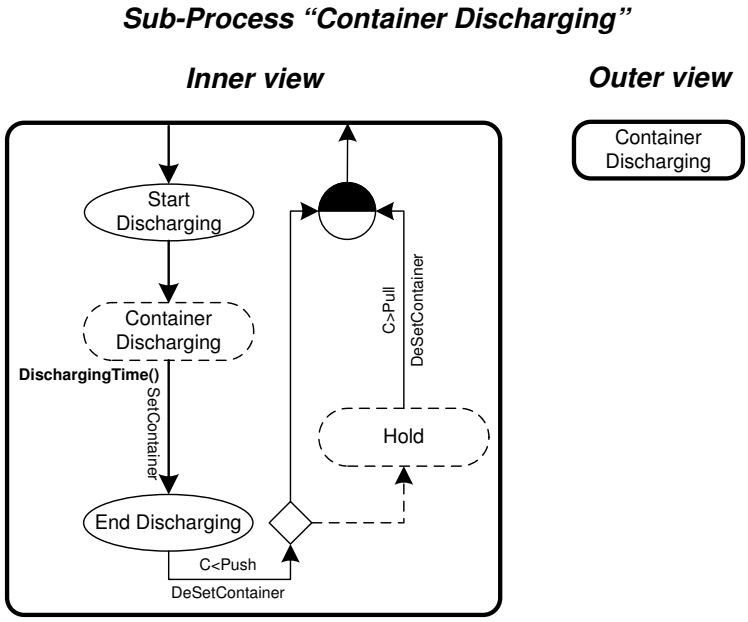


Figure 1.32: The sub-process "Container Discharging" used in the EAD of the Crane model object.

queues that implies the development of a push/pull job transfer system among ACs (or CCs): this transfer system has been programmed into the components behavior by i) requiring acknowledgment messages to confirm at the sender component if the job sent to an output port is definitely accepted by the receiver component, and i) sending to backwards components a request message for pushing out a job (if any job can be sent). Obviously, also the continuous communication among components to take decision on the use of shared passive resources takes a role in complicating the behavior of the ACs. In Figure 1.25 the HCFG model of the quay crane operations model is presented. In opposition to modern VIMS, this system is quite complex to analyze, due to the high number of channels among ACs. A more compact representation of the model may be obtained using set of identical instances of the same component type. This is a fine solution for model scaling, but it also complicates the model understanding.

The set of ACs used to compose the HCFG model of the system is the following:

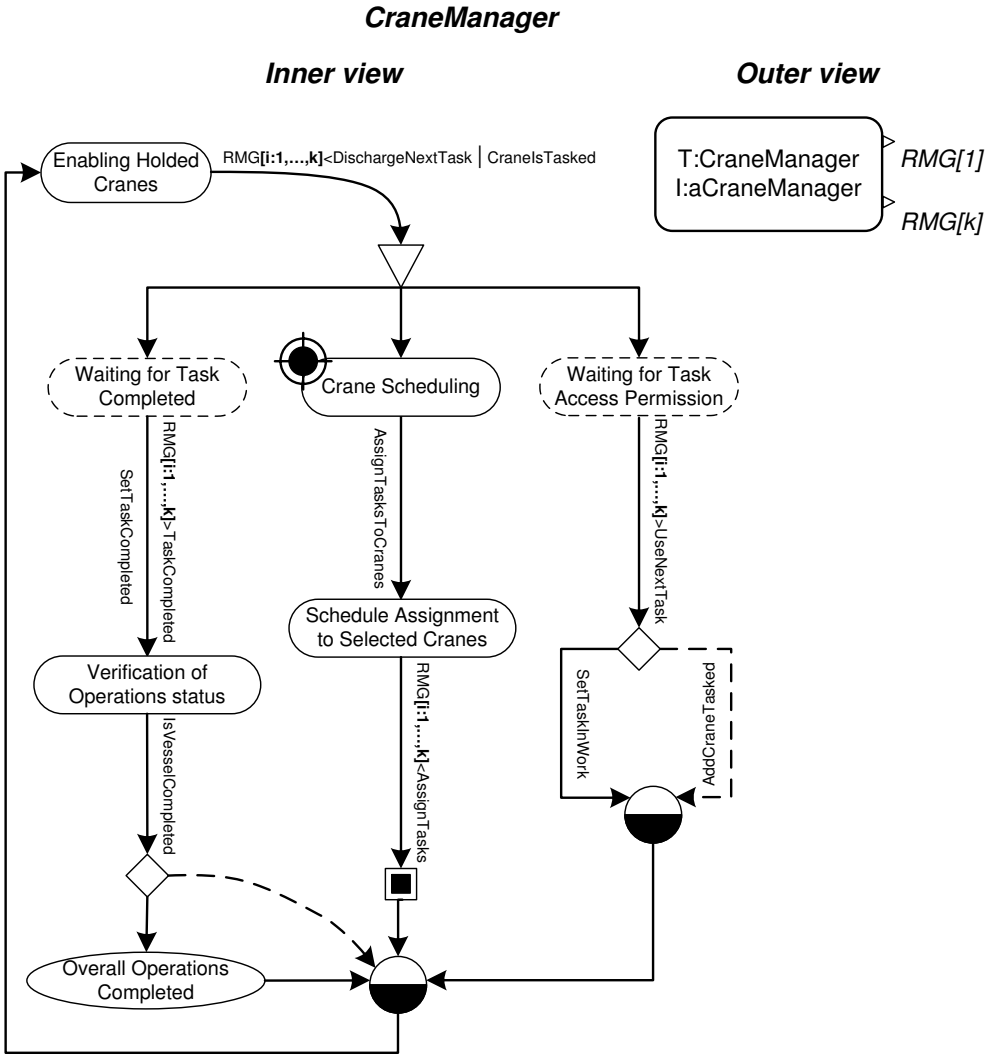


Figure 1.33: The EAD and the corresponding outer view of the CraneManager model object.

i) a *Crane* (Figure 1.26) for depicting the QC resource, *ii*) a *CraneController* (Figure 1.27) to manage the interaction among Cranes that are competing to discharge tasks under specific constraints and assign the sequence of tasks of the berthed vessel, *iii*) a *FiniteQueue* (Figure 1.28) to depict the finite buffer area under each quay crane, *iv*) a *ContainerLoader* (Figure 1.29) to perform container set-up operations in each straddle carrier (here intended as a job able to carry on containers), *v*)

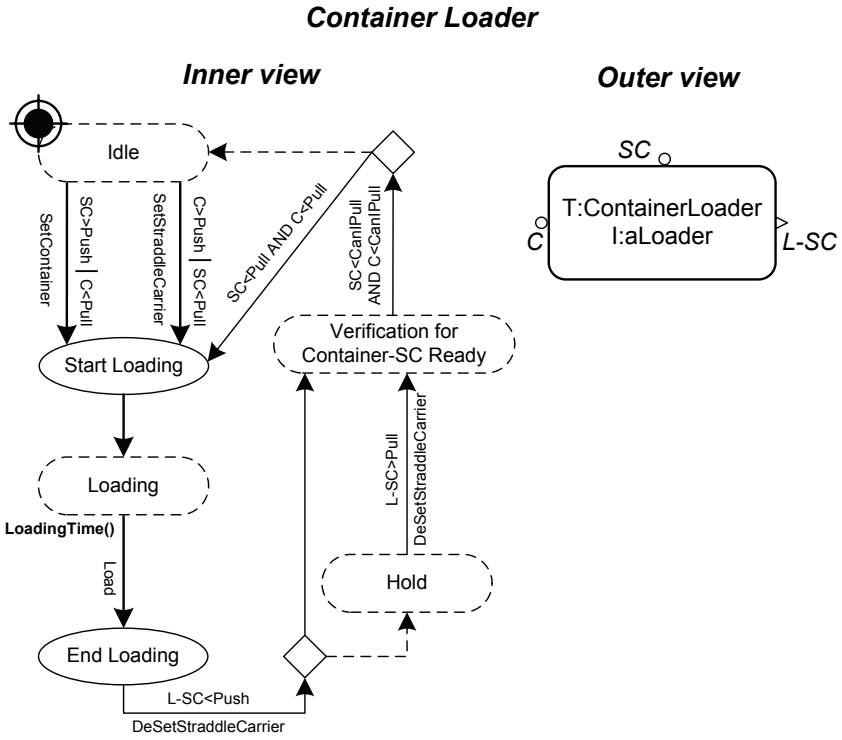


Figure 1.34: The EAD and the corresponding outer view of the model object ContainerLoader model object.

a *ThinkingStation* to depict the time required by each loaded straddle carrier for transferring the container from the quayside to the yardside, set-down the container and come back empty under the crane area. We omitted to depict the ThinkingStation, nevertheless it is a CC presented by Sargent [78] composed by a set of identical servers (one for each straddle carrier) and by a controller that assign incoming jobs to idle servers: Sargent called this CC the *KParServers*.

Finally, we provide a representation of the quay crane operations model using a HMP model, as shown in Figure 1.30, based on the coupling of six types of model objects and a sub-system. The model objects are: *i*) a *Crane* (Figures 1.31 and 1.32) for depicting the QC resource, *ii*) a *CraneManager* (Figure 1.33) to manage the interactions among QCs working on the same vessel and assign the sequence of tasks of the berthed vessels, *iii*) a *ContainerLoader* (Figure 1.34) to perform container set-up operations for empty SCs, *iv*) a *Queue* (Figure 1.35) to depict the waiting

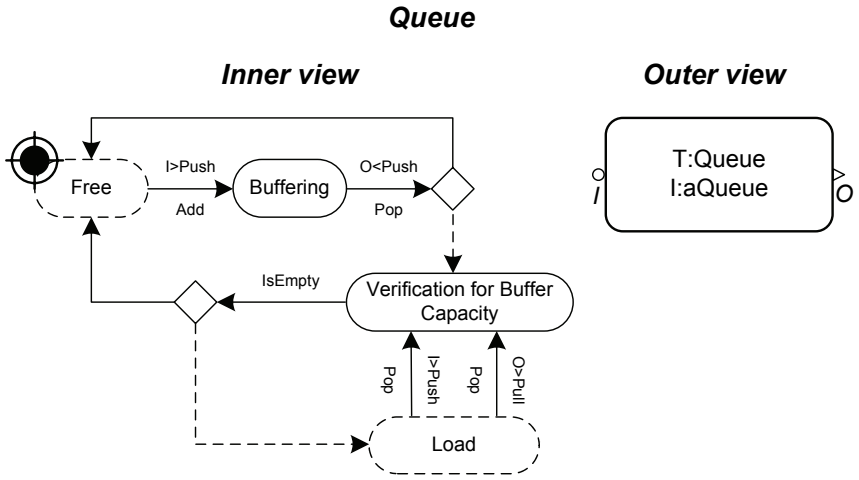


Figure 1.35: The EAD and the corresponding outer view of the model object Queue model object.

line of empty SCs that wait for load discharged containers and v) a *FiniteQueue* (Figure 1.16) to depict the buffer area under a crane (for no more than 6 TEUs).

With this approach, a modeler can design the behavior of a set of context specific model objects (e.g., cranes and queues), even unrelated, and then he/she can reproduce a certain system by linking these model objects and simply specifying suitable parameters (e.g., the distribution function for the crane discharge time). If the designed model objects are collected in a library in a "point & click" simulation environment, they are close at hand to be reused to design a new system.

It is clear that by using this approach we provide a compact way of representing the system logic: in fact, entities flow through the model objects and the relationships between the model components are very clear. This is true especially if the model end-user looks at name of the ports, which explicitly identifies which are the input(s) and the output(s) of each model object.

Moreover, if one wishes to further simplify the readability of the model by hiding a given part of the system, this can be performed in a simple way by grouping and depicting the selected system part as a sub-model. For instance, in Figure 1.30 the *ContainerStoraging* is a sub-model that receives in input a loaded SC and returns an empty SC after a while (the yard cycle time); focusing our attention on the quayside

operations, we are not interested in understanding how an SC has been unloaded and why it has returned after a certain time period, therefore the use of a sub-model in this context appears convenient.

The behavior of each model object is designed by using an EAD. Each EAD is composed by *i*) default components (events, activities, logical nodes, edges) and *ii*) context specific timers and functions. Both timers and functions must be developed around a set of global variables related to each process (e.g., a list of entities for the infinite queue depicted by the model object Queue).

This approach has its strong point on the use of functions to check conditions and execute algorithms devoted to take decisions. For instance, once a vessel arrives at berth, the CraneManager uses a function that solves a mathematical model (the Quay Crane Scheduling model proposed in Chapter 3) to dynamically assign the tasks of the vessel to a specific set of cranes. Obviously, this approach is possible using our MP or EGs and HCFGs. Petri Nets are a powerful tool for modeling a system at a very low level, describing also conditions and complex constraints by means of its intuitive formalism, but a PN cannot be re-designed at runtime in case of need (e.g., is not possible to use the PN in Figure 1.24 to simulate the discharge process of a vessel with a different structure).

1.7 Simulation and Optimization of Logistic Systems

A modern simulation platform for logistic systems must provide an easy-to-understand language for system modeling, but also a tool for designing of optimization problems.

For us, a platform for the optimal management of logistic systems must be based on the following three phases: *i*) *system modeling*, *ii*) *model analysis* and *iii*) *system optimization*. In the first phase, the real system is modeled by using the appropriate MP.

Numerical results from the simulation model obtained in the system modeling phase are analyzed in the second phase by means of a statistical analysis tool. Performance measures are evaluated by the outputs of a set of simulation experiments

of the simulation model. In the model analysis phase, indices and parameters are defined on the simulation model by means of some tool or language. The evaluation of these indices and parameters provides the set of performance measures that is approximately necessary to improve during the optimization phase.

Thus, in the third phase an optimization problem is defined. General problem setting is made of input and output variables, objective function and constraints. The nature of the optimization problem is intrinsically stochastic (due to the nature of the simulation output variable).

Simulation modeling and analysis should become an ordinary P&C tool, e.g., to evaluate the performance of the container terminal when changes occur in the system configuration (*what-if analysis*) or to use simulation-based optimization techniques to optimize the whole system [35].

Therefore, the Chapter 2 is devoted to show the feature of a simulation technology known as simulation-based optimization, while the Chapter 3 describe how the case study proposed in Section 1.8 is optimized using this new methodology.

1.8 Conclusions

In this Chapter has been proposed a Modeling Paradigm to support the development of simulation models oriented to the optimal management of logistic activities. The MP uses a holistic approach to capture the complex relationships among sub-systems and allows for a direct representation of the hierarchical structure of the decision making process for system management. System modeling is completed by a flow-chart based definition of processes involved in the model at hand. Real case examples of modeling have been presented, comparing our approach to those provided by Event Graph models, Petri Nets and Hierarchical Control Flow Graph models, with the aim of supporting the future design of simulation-based optimization techniques.

Chapter 2

Simulation-based Optimization

The need for the optimal management of real-systems is well recognized since some decades, as for instance in *manufacturing*, *logistics* and *defense*. In this context, simulation is widely adopted for an accurate modeling of real systems, evaluating their performances and analyzing the effects of alternative conditions and organizational policies. General purpose and problem-oriented software packages have increased the diffusion of simulation as analysis tool in system management. Very challenging for real applications should be a new methodology known as *Simulation-based Optimization*, in that it combines both simulation and optimization techniques with the objective of optimizing the performance measures of a system under study. Some optimization features have added in the majority of commercial simulation software, but they are not usually customizable. Furthermore, those features just play around the possibility of optimizing an objective function under box constraints.

This chapter discusses how optimization uses simulation to design new systems and to improve the performance of existing ones. Therefore, we present an overview of *simulation-based optimization*. In particular, we focus on those techniques that select new system configurations by means of suitable *moves*, i.e. using procedures that change the values of key variables in an intelligent way. Thus, we discuss about the importance of using intelligent moves, which collides with the use of commercial simulation-based optimization packages. Then, we introduce two frameworks newly developed for both global and local search. Finally, a comprehensive example of the performance optimization of a production line will be given to show the effect of the use of advanced selection rules.

2.1 Introduction

There is a wide range of decision-making problems (e.g., *transport*, *storage*, *manufacturing*, *medical* and *military applications*, *demand management*, etc.) whose efficient and effective management has a key role in modern economies. All of these problems involve some combination of complex dynamics, uncertainty, high-dimensional decision vectors and a need to make decisions that take into account the impact on the future.

Decision-making problems are studied, with particular kinds of stochastic optimization problems, for the development of convenient decision support models. In the beginning these problems were essentially of types that could be formulated suitably by using scenarios. Subsequently they were reduced to large, structured, deterministic problems [25] generally solved using techniques based on *optimization*. On the other hand, more recently many of those problems characterized by uncertainty, which could not be adequately handled by optimization, have required the use of simulation to achieve feasible and practical solutions: in fact, through the decades simulation was verified to be very effective and in many cases, even the unique, practical choice to approach decision-making problems.

Several decision problems in manufacturing, goods transportation and storage, are primarily studied using *discrete-event simulation* (DES). Over the decades, many *simulation platforms* (SPs) have been developed with the aim of supporting the modeler in the design, testing and evaluation of simulation models. More remarkable commercial SPs are *Arena*, *AweSim!*, *exteNd*, *ProModel* and *WITNESS*. Seila [81] defined a *simulation platform* (SP) as a software environment used to develop, test and run a simulation experiment. The following list provides the minimal capabilities that must be available in a SP:

1. *Model representation.* A tool to describe a system using mathematical and logical relationships among system objects (e.g., a modeling paradigm, as proposed in the Chapter 1).

2. *Distributions set and pseudo-random numbers.* A tool for pseudo-random number generation to support the Monte Carlo generation of random deviates from several distributions.

3. *Simulation methodology.* A tool to reproduce the system behavior over the time and execute statistical computations (e.g., confidence intervals by batching elementary observations).

Law and Kelton [35] provided a list of advantages of a SP: *i*) reduction of programming effort and cost by means of a collection of modeling features, *ii*) simplification of model modification, *iii*) better error detection for simulation-specific errors and *iv*) faster model analysis via statistical tools. They also classified modern SP in *general-purpose* and *application-oriented*. Following Pidd [65] the majority of commercial simulators may be classified as Visual Interactive Modeling System (VIMS) – e.g., *Arena* and *exteNd*. VIMSs allows the modeler to develop a model using a flow chart based methodology to depict the logic of the system, which is then translated by the program to generate the underlying model code. VIMS are successful because they offer the prospect of rapid application development by people who are not computer professionals [66].

As we stated before, almost the totality of the decision-making problems arising from a real system whose dynamics is affected by uncertainty are faced by resorting to two widespread approaches: *optimization* and *simulation*. While optimization can allow the modeling of the whole system from a static, deterministic point of view, with a significant loss of information, simulation allows to keep a more realistic, dynamic and non-deterministic point of view on a system without a significant loss of information. Therefore, it should be preferable to develop decision support models for non-deterministic systems that are based first on simulation and, after that, possibly also on embedded optimization models. Alternately, one stimulating possibility is that of combining both simulation and optimization techniques to pursue the so called "optimum seeking by simulation" [35]. The possibility of the optimization of simulation models have made possible by the increasing availability of computing power and memory over the last two decades [26]. This development offers the opportunity of automating the search for the "optimal" values for the controllable inputs of the simulation model [8]. Different approaches have been proposed to optimize a simulation model – primarily metaheuristics – [5], and many *optimization packages* (OPs) have been developed and integrated in major commercial SPs (e.g., *OptQuest* for Arena).

Thus, we claim that a modern SP for the optimal management of large and complex systems must include *simulation-based optimization* (SO) techniques – also known as *optimization via simulation* or *simulation-optimization* methodologies.

The Chapter is organized as follows: next section provides a brief overview of simulation-based optimization. The subsequent section focuses on the relevance of the system configurations selection in a large space state. Successively, two recent SO frameworks are introduced. Finally, a production line example shows the importance of the use of an *ad hoc* methodology for system alternatives selection.

2.2 An overview on simulation-based optimization

SO is the most important new simulation technology in the last two decades [36]. SO is the optimization of performance measures based on outputs from stochastic (primarily discrete-event) simulations [24, 27]. Thus, an optimization model is defined upon a simulation model, afterward a SO technique searches the optimal system configuration within the defined feasible region using a simulation routine.

As in any optimization problems, there are the following primary components [88]:

- deterministic input parameters and stochastic output variables from the DES model;
- objective function;
- constraints.

Input parameters (or *controllable parameters*) are defined over a *feasible region* Θ, while output variables, which are function of input parameters, are estimated by mean of simulation. Since output variables are system performance measures, so they are *quantitative* in nature. Despite classical mathematical problems, the input parameters may be either *qualitative* (e.g., queue disciplines) or *quantitative* (e.g., the number of type of machines for a particular objective). Quantitative input

parameters are distinguished between two cases: the continuous values and discrete values.

The objective function combines output variables and at times quantitative input parameters (e.g., the number of buffer spaces in a just-in-time production line). The goal is to determine input values such that objective function is optimized.

Fu [24] separates constraints in *implicit* versus *explicit*, and *deterministic* versus *stochastic*. An implicit constraint is something like "the number of machine tools in a numerical control machine cannot exceed 10", whereas an explicit one would be more like "the number of machine tools (summed over all the numerical control machines) in the whole system cannot exceed 100". These are also deterministic constraints, while a stochastic constraint might be "the proportion of parts having a flow-time greater than 30 minutes should not exceed 5%".

Thus, SO is concerned with solving the following problem:

$$\min_{\theta \in \Theta} f(\theta) = E[L(\theta)] \tag{2.1}$$

where $f : \Theta \rightarrow \mathbb{R}$ is an objective function that is subject to noise and Θ is the feasible region (that is, for any feasible point $\theta \in \Theta$, $f(\theta)$ cannot be evaluated analytically). $L(\theta)$ is a random variable which depends on the parameter $\theta \in \Theta$, and is a simulation estimate of the output of a system configuration (i.e. the "point" or "solution" θ). $L(\theta)$ is referred as the *sample performance*. Thus, $f(\theta)$ is an expectation of some random estimate of the performance of a complex stochastic system given a parameter θ.

The previous problem can be also in the form of maximization of $f(\theta)$. Our interest is particularly in solving the problem (2.1) in situations where the objective function value $f(\theta)$ at any θ cannot be evaluated exactly, but need to be estimated using simulation.

2.2.1 Simulation-based optimization generic scheme

Law and Kelton [35] present a clear scheme of the interaction between optimization and simulation. Here we present a more detailed scheme, with the purpose to

underline the simulation dependency of the optimization in the system optimization process and show the typical data structure used in a SO framework (Figures 2.1 and 2.2).

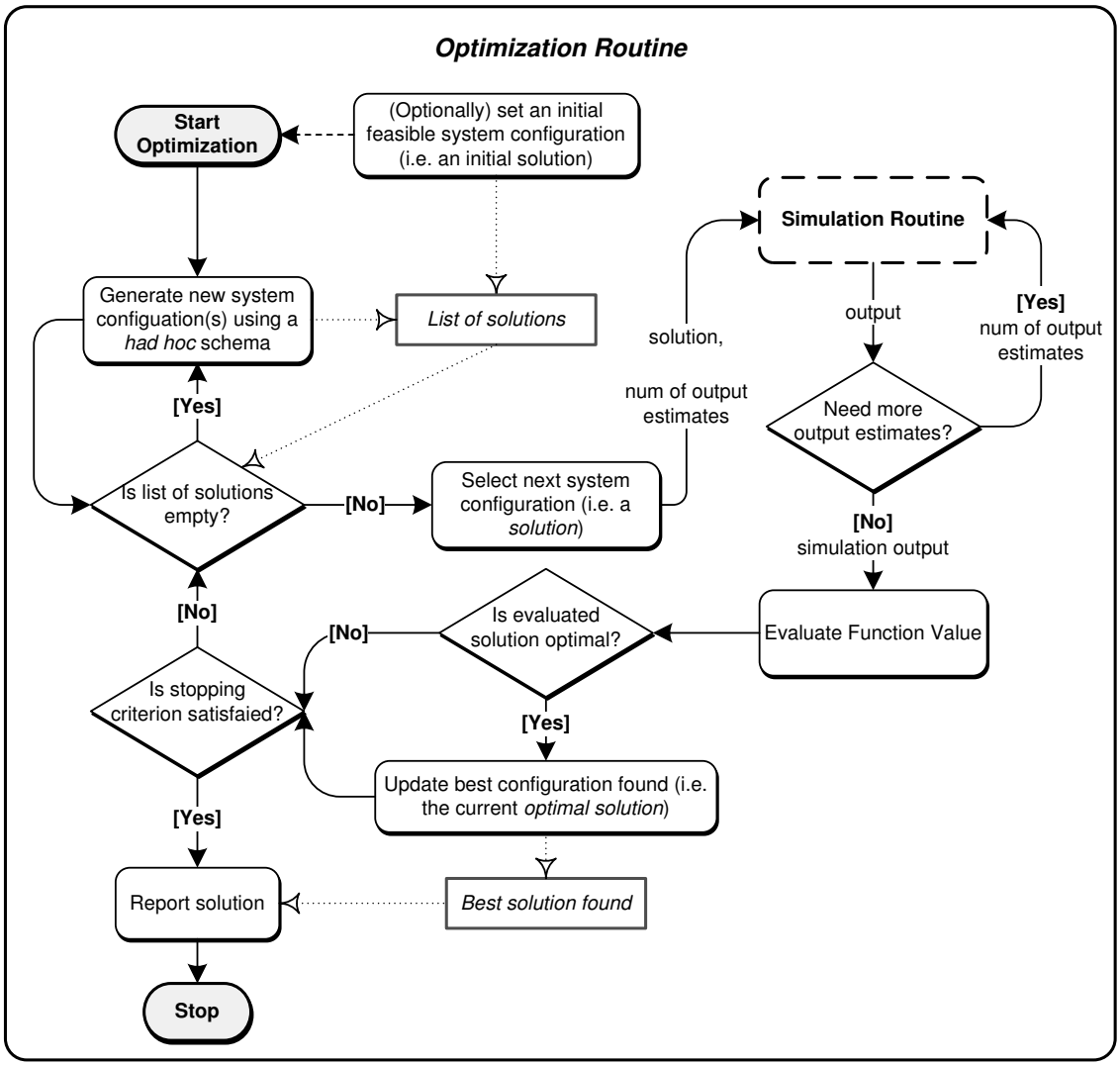


Figure 2.1: System optimization via simulation.

The steps in the general scheme in Figure 2.1 are extended by the selected framework for optimization via simulation (in Sections 2.4 and 2.5 are reported two late search schemes). Some issues in the design of all the SO frameworks concerns the steps related to i) the updating of the best solution found (i.e. the *estimation of*

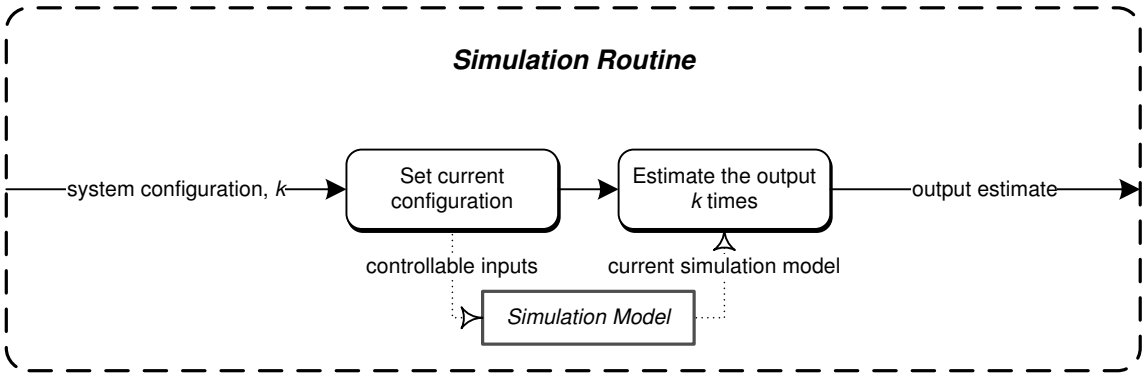


Figure 2.2: Simulation routine.

the optimal solution), and *ii*) the determination of the additional output estimates for a specific solution. Both issues are challenging in the SO setting due to the noise in the estimated objective function values.

About the first issue, commonly, the most considered estimates of the optimal solution in stochastic SO are [2, 3] *i*) the solution with the best estimated objective function value, *ii*) the most visited solution, and *iii*) the solution with the highest estimated objective function value among solutions that have been visited sufficiently often. Obviously, only the first estimates is computationally practicable for large feasible regions: in fact, the other estimates reveal the necessity to memorize several (if not all) the solutions evaluated during the search process.

As regards the second issue, the variance of the observations from the simulation replications is not known in advance. Therefore is required a methodology for computing the number of replications required to produce a good estimate of the simulation output. Some methodology have been developed [51, 14, 13], but still the celebrated *Rinott's procedure* [72] is usually used to prescribe how many simulation experiments are needed for each solution. These methodologies typically assume that is possible to collect *independent and identically distributed* (i.i.d.) normal observations from a each solution. Generally, these normal observations are simply the batch means from one long run of each solution, or the sample means from independent replications of the same solution.

The use of Rinott's procedure for the determination of the number of estimates

needed to evaluate a solution θ, is recommended in SO to reduce the presence of stochastic errors in the estimated objective function values. Therefore, in the following we report this procedure that will be integrated in the metaheuristic schemes adopted within this PhD thesis. Finally, note that estimation is usually not incorporated explicitly in SO methods, above all to preserve the structure of search schemes for deterministic optimization that are adapted for stochastic SO.

Determination of the number of estimates using the Rinott's procedure Rinott's procedure makes the following assumptions: there are $k \geqslant 2$ solutions, X_{ij} is the j-th independent observation from solution θ_i, $X_{ij} \sim N(\mu_i, \sigma_i^2)$, where μ_i is the expected value of the output from the solution θ_i and σ_i^2 is the related variance (with μ_i and σ_i^2 unknown), and all the X_{ij} are allegedly i.i.d.. In the Rinott's procedure: h is the *Rinott's constant*; n_1 is the number of observations X_{ij} for each solution θ_i; $1-\alpha$ is the probability of correct selection of the best solution, whenever the true best is at least an amount δ better than the others. Therefore, δ represents the minimum detectable difference between competing solutions and is called *indifference-zone parameter*. Both α and δ are used defined, with α small. The Rinott's parameter h depends on n_1, α and the number of comparing alternatives k. The values of h can be found in the tables in [92].

To implement this simple procedure, the following steps are taken.

Initialization Select confidence level $1-\alpha$, indifference zone parameter $\delta > 0$, and the common first stage sample size ($n_1 \geqslant 2$). Obtain Rinott's constant h.

Stage 1 For each solution θ_i, with $i = 1, \ldots, k$:

1. Obtain n_1 observations X_{ij}, with $j = 1, \ldots, n_1$.
2. Compute the sample mean:
$$\overline{X}_i(n_1) = \frac{\sum_{j=1}^{n_1} X_{ij}}{n_1}.$$
3. Compute the sample variance:
$$S_i^2 = \frac{1}{n_1 - 1} \sum_{j=1}^{n_1} \left(X_{ij} - \overline{X}_i(n_1)\right)^2.$$

4. Compute the final sample size:

$$N_i = \max\left\{n_1, \left\lceil \left(\frac{h \cdot S_i}{\delta}\right)^2 \right\rceil\right\}.$$

Stage 2 For each solution θ_i, with $i = 1, \ldots, k$:

1. Take $N_i - n_1$ additional i.i.d. observations from solution θ^i, independently of the first-stage sample.
2. Compute the overall sample mean:

$$\overline{X}_i(N_i) = \frac{\overline{X}_i(n_1) \cdot n_1 + \sum_{j=n_1+1}^{N_i} X_{ij}}{n_1 + N_i}.$$

2.2.2 Classification of the simulation-based algorithms

Banks et al. [6] classified the main approaches in SO according to algorithms that: *i*) guarantee asymptotic convergence to the optimum (generally for continuous-valued parameters); *ii*) guarantee optimality under deterministic counterpart (i.e., if there were no statistical error or sampling variability; generally based on mathematical programming formulations); *iii*) guarantee a pre-specified probability of correct selection (generally from a pre-specified set of alternatives); *iv*) are based on robust heuristics (mainly combinatorial search algorithms that follow evolutionary strategies, e.g., genetic algorithms).

Fu [23] divides the techniques used in SO into the following main categories: *i*) **statistical procedures**, e.g. *sequential response surface* methodology, *ranking & selection* procedures, and *multiple comparison* procedures; *ii*) **heuristics**, i.e. methods directly adopted from deterministic optimization search strategies, i.e. *meta-heuristics* (such as *simulated annealing*, *nested partitions* and *tabu search*), and *evolutionary algorithms* (for instance *genetic algorithms*, the *scatter search algorithm*, etc.); *iii*) **stochastic optimization**, e.g. *random search*, *stochastic approximation*; *iv*) *others*, including *ordinal optimization* and *sample path optimization*.

Otherwise, Ólafsson and Kim [58] classified the techniques used by SO considering the nature of the feasible region. They classified Θ as *i*) *infinite and uncountable* if the decision variables are continuous, *ii*) *finite and fairly small*, if the decision variables are discrete and Θ consists of about 30 alternative solutions (or less), or *iii*)

finite but combinatorially large, if the decision variables are discrete and the feasible region is made up by a number of combinatorial solutions. In the first case, Ólafsson and Kim [58] suggest to use the *stochastic optimization*, e.g. using the *stochastic hill climbing* method. In the second case is possible to examine all the available system configurations, therefore the use of *statistical procedures* is recommended. In the latter case, considering the high number of alternative system configurations, we know that the evaluation of all the alternatives is not practicable: in fact, generally we will need to make n independent simulation for each system configuration followed by the use of the sample mean over the n replications as an estimate of the expected value of the objective function corresponding to a configuration. Then the use of *metaheuristic* is commonly the most appropriate approach.

Thus, our purpose concerns problems in which there are only discrete input parameters and there are a combinatorial number of alternatives (which means that is not possible to evaluate all the possible alternatives). Therefore, in the following we focus our interest on the metaheuristics approach to the SO. In this perspective, the selection of promising system configurations holds a key role in the identification of the optimal alternative with desirable convergence properties.

2.3 Selection of Promising Solutions: Commercial Packages or Ad Hoc Software?

Advanced OPs for discrete-event simulator are designed to search for optimal solution to the following class of optimization problem [88, 4]:

$$\textit{minimize} \quad \text{or} \quad \textit{maximize} \quad f(\theta) \tag{2.2a}$$

subject to

$$g(\theta) \leqslant 0 \qquad \text{(Constraints)} \qquad (2.2b)$$

$$h_l \leqslant h(\theta) \leqslant h_u \qquad \text{(Requirements)} \qquad (2.2c)$$

$$l \leqslant \theta \leqslant u \qquad \text{(Bounds)} \qquad (2.2d)$$

The objective function $f(\theta)$ combines the "variables" $\theta \equiv (\theta_1, \ldots, \theta_n)$ (i.e. combines the expected values of the output variables and *in case* quantitative input parameters).

The set of constraints must be linear, i.e., $g(\theta)$ is a linear function. The requirements are simple upper and/or lower bounds imposed on a function $h(\theta)$ that can be linear or non-linear. The values of the bounds h_l and h_u must be known constants. The whole of θ are continuous or discrete (or *both*) and must be bounded.

As reported in [36], the most notable OPs are *AutoStat*, *exteNd Optimizer*, *WITNESS Optimizer* and *OptQuest*. AutoStat and exteNd Optimizer are OPs that use evolution strategies. The WITNESS Optimizer uses metaheuristics approaches, in particular it is based on the simulated annealing and tabu search methods. The OptQuest package is the most widespread (adopted for instance in the simulators *Arena* and *Simul8*) and probably the most complete: its optimization engine is based on the use of metaheuristics (in particular the tabu search algorithm), evolution strategies (such as the scatter search) and artificial neural networks.

At this point, we are interested in explaining how a modeler must optimize a simulation model interacting with a commercial OP integrated in a SP. Therefore, we briefly illustrate the main feature of the OP developed by the *OptTek Systems, Inc.*, in the version that has been developed to be tightly integrated within the *Rockwell Automation, Inc.*'s *Arena* (version 9.0) SP: **OptQuest**.

As the above general description of the *SO* approach should have suggested, the two key phases in attempting to use an OP to choose the optimum values of the input parameters for a system that has been modeled in a SP are:

- *Selection of the input parameters*. This phase concerns the selection of a subset of input parameters whose values will be changed by a SO technique under specific constraints; the SO technique aims to find a combination of values

that produces the best objective function. Each set of input values correspond to a unique system configuration, or rather to a *solution*.

- *Identification of the objective*. In this phase the modeler must identify the goal and through the SO technique evaluate the quality of each alternative solution. The goal is the maximization or minimization of an input dependent mathematical expression.

Afterwards, we now can see how OptQuest for Arena allows the modeler to attempt the two phases for the optimization model definition. Here we refer to information extracted by some applied research study [73] and the "OptQuest for Arena User's Guide" [9]. In the first phase, the modeler must specify the controls (i.e., the controllable parameters associated with the system being modeled) that OptQuest is allowed to select values for, establish upper and lower limits for each control, and define linear and non-linear constraints (non-linear constraints include output variables of the simulation model, therefore non-linear constraints feasibility is checked after the evaluation through simulation of the solution). The set of controls the modeler is allowed to select from includes all Arena model user defined variables (e.g. the number of completions of workpieces in a production line) and all model resource capacities (e.g. the number of servers in a service station) – a snapshot of the OptQuest GUI for Arena is shown in Figure 2.3. In the latter phase, the modeler must define the objective function that combines any statistic defined in the model, i.e. typically, system performance measures. (In the User's Guide is heedless to develop complex objectives, despite the solver can approach also very complex objective functions).

As it easy to recognize, using this kind of SO packages is not possible to approach many interesting and practical optimization problem (for instance, see the Chapter 3 at page 97). In fact:

In a previous study on an OP developed as embedded tool for an open-source simulator [46], we have shown that an objective function, sometimes, includes both input and output parameters. Then, we have evaluated a cost-function for a production line, whose costs where dependent by the line throughput, the average flow-time,

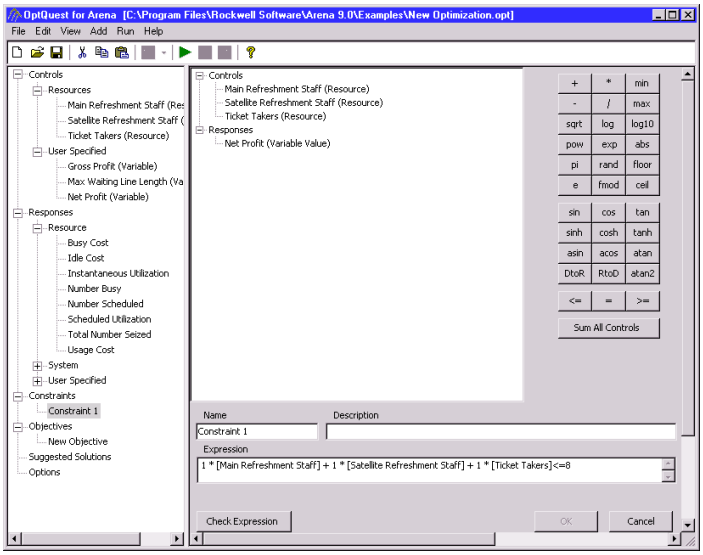


Figure 2.3: The OptQuest GUI for Arena.

and the capacity of the buffer areas located after and before of each machine. Obviously, the first two elements of the objective function were performance indexes (i.e. outputs by the simulation model), while the buffer capacity was a system parameter (i.e. input to the simulation model). The value of the buffer capacity was obviously also included in the optimization model constraints in order to change their values.

1. Commercial OPs let the modeler to define an optimization problem using the model output and quantitative input parameters. Nevertheless, sometimes may be interesting to search for a set of values of both quantitative and qualitative input parameters that maximizes (or minimizes) the performance of the system.

2. Optimization problems in some closed systems may refer to a static set of model entities (e.g., the vessels in a maritime container terminal), whose parameters may be used to perform some assignment or scheduling at the system resources.

3. The set-up of large optimization problems (i.e. with many variables and constraints) is a manual labor that may require too much effort for the modeler

and it may lead mistakes in copying. Furthermore, storing the structure of a large optimization problem usually requires too much computer memory: in these cases, is preferable to use a lightest form to memorize the structure of a problem, e.g. memorizing only the variables whose values cannot be given by others and using ad hoc algorithms to evaluate the feasibility of each solution.

In conclusion, we believe that commercial SO packages are not yet enough mature to approach many interesting problems in a satisfying way. Therefore, the approach of certain SO problems needs the use of ad hoc software or customizable OPs. As we anticipated in the previous section, using metaheuristics is the most appropriate approach in the context of SO for finite and combinatorially large feasible regions. Then, in the following we discuss on how metaheuristics are generally adopted in an environment for optimization via simulation.

A metaheuristics is a high-level strategy that guides other heuristics in a search for the optimum solution in a feasible region. Typically, metaheuristics searches for the optimal solution through the feasible region performing *global search*, *local search* or both.

SO metaheuristics based on global search, generally starts from the examinations of a set of randomly sampled points from the whole feasible region to address the search at the most promising subregion. In this case, if an initial feasible solution exists (e.g., the actual real system configuration), it may be compared at the end of the search with the (sub-)optimal solution or it may be evaluated with one of the randomly chosen feasible points. The previous, are generally *memoryless* metaheuristics, i.e. methods that rely on semirandom processes that implement a form of sampling. Examples of memoryless methods include semigreedy heuristics and the prominent "genetic" and "annealing" approaches inspired by metaphors of physics and biology. Metaheuristics *with memory* are provided of *rigid memory* or *adaptive memory*. This metaheuristics search through the whole feasible space by selecting solutions and avoiding the exploration of un-promising or yet explored sub-regions. Examples of methods with rigid memory ar typical metaheuristics which implements branch and bound strategies, while the most notable metaheuristics that uses adaptive memory is Tabu Search [28, 29]

If a metaheuristics for simulation-optimization is based on local search, then its

search strategy involves moving successively between neighboring feasible points in search of the optimal solution [1]. Thus, in this case the metaheuristics searches for optimal solution within the feasible region by determining and evaluating alternative solutions into the neighborhood of the current solution, eventually replacing the current solution with a neighbor. In SO, the neighborhood is a selected set of alternative system configurations classified as proximal to the current configuration. Neighbors are classically generated using the so called *moves*.

First of all, let us provide a clear and simple definition of move. As stated in [48], a *move* is a procedure that generates a new feasible solution θ_j similar (near) to the current candidate solution θ by changing just one or (at most) some values of θ, namely some values of the input parameters of the simulation model that is referred to the current candidate solution (the use of such a distance measure may be helpful, if not crucial, in the generation of neighbor solutions). At the moment we do not consider the generation of un-feasible candidate solution (e.g., this may occur in the Tabu Search metaheuristic, where the generation of solutions that fit only a subset of the problem constraints is allowed). Therefore, the problem of correctly defining the neighbor solutions concerns first the definition of the set of moves and successively the selection of the move that must be performed – this is the so called *neighborhood generation problem*.

Often a move is a simple procedure that increases or decreases, with the same probability, the value of only one input parameter per time [7]. The value is increased (decreased) of a quantity that do not go beyond the fixed bounds. Rarely the new value is chosen randomly between the minimum and maximum value of the parameter. In both cases, the new value for the selected input parameter must not violate the constraints of the mathematical model (2.2) and, if possible, should drive the search process towards a feasible solution with a better value of the objective function. The "classical" move used in [7] changes a single parameter value per time, but this could reveal as a not efficient choice in some strictly bounded mathematical models of our interest, where a strong dependency has been detected among the input parameters of the SO model. A better choice should use moves which consider the above dependencies.

Our proposal is to construct a neighborhood generation procedure that once a

parameter p has been changed, directly reconsiders the values associated to other parameters strongly dependent by p, which belong to the same constraints and requirements. This is expected to outperform the classical approach where parameter values are changed one at a time.

In Legato Trunfio and Mari [48] the new proposal has been pursued in a framework of production logistics and numerical evidence was given on the better performance of the new neighborhood generation procedure. Specifically, it gives the possibility to evaluate a larger number of feasible solutions and to directly discard solutions that are clearly not worth to generate. Recalling that the solution evaluation is a time-consuming step in a simulation optimization algorithm, one may easily recognize that the feature of avoiding usefulness evaluations is particularly appreciable. Finally, a Tabu Search scheme was embedded in the SO algorithm, to implement the neighborhood generation procedure. Details are given in [48].

Numerical results reveals that, to a large extent, the success and the efficiency of the entire search process relies upon the right balancing of two phases the *exploration* of the whole feasible region and the *exploitation* (i.e. the intensification process, in the tabu search language) of the neighborhood of a promising solution θ.

In the Sections 2.4 and 2.5 are described two recent metaheuristic frameworks for the exploration and exploitation of Θ. Both framework use randomization, that is a popular and effective approach to escape local optima. In the following, the *Nested Partitions* method is shown, while in subsequent Section, two random-methods based on the *Balanced Explorative and Exploitative Search* framework are described.

2.4 Nested Partitions

In this Section is described a randomized method for solving global optimization problems, called *Nested Partitions* (NP) and proposed by Shi and Ólafsson [56, 83, 84]. The NP method iteratively partitions the feasible region in a promising subregion and in the corresponding surrounding region, with the goal to identify the smallest most promising subregion that may include the optimal solution. The

NP algorithm evaluates each promising region by sampling a set of points included in the region, and then defining the average value of the objective function of this points (that is an estimate of how much promising is the examined subregion). Shi and Ólafsson [83] stated that the method is shown to converge with probability one to a global optimum in finite time.

This metaheuristics is inspired by the popular *branch-and-bound* (B&B) algorithm [34]. Note that this algorithm is a non-randomized optimization method originally proposed to solve integer linear programs, and generally adopted for global optimization [30]. The basic idea behind B&B is to systematically partition the feasible region and estimate upper and lower bounds for each of these subregions, as well as for the entire feasible region. The bounds obtained for each subregion are compared with the bounds for the entire feasible region. This comparison is used to eliminate each of the subregions until only the optimal solution remains.

The NP method systematically partitions the feasible region into subregions, estimates the potential of each region by random sampling a set of included feasible points, and then concentrates the computational effort in the most promising region. Therefore, NP method combines both global and local search by exploring the whole feasible region and exploiting the most promising region. Like the B&B, the NP method aims to find *singletons* regions, i.e., subregions with that include only one solution.

The NP method has been introduced for deterministic optimization, but it can also be applied to stochastic problems [57, 63]. In fact, every time a point is sampled by a subregion, it can be evaluated using such estimator of the performance of the point (e.g., by using simulation in a SO approach).

2.4.1 Method description

Formally, the feasible region Θ is a finite set of n-dimensional points, therefore $\Theta \subset \mathbb{R}^n$, such that:

$$\Theta = \{\theta = (\theta_1, \ldots, \theta_n) \,|\, \theta_i \in \Theta_i\},$$

where Θ_i is a finite set of values that θ_i can assume, and the cardinality of Θ_i

may depends on the region that is currently partitioned. Let σ be the region that the NP aims to partitions by considering the values that θ_i can assume in σ. Thus, let $M_i(\sigma)$ be the number of possible values of θ_i in σ, it is definite by a function $m(\cdot)$ such as $M_i(\sigma) = m(\theta_i, \sigma)$.

In the following, the main phases of the NP scheme are briefly described [83].

- *Partitioning.* At each iteration, the attention is focused on the decision variable θ_i, and Θ is partitioned into *i*) a number of subregions which depend on Θ_i, and *ii*) in the surrounding region of all the previous subregions (if one exists). This subregions cover the feasible region but concentrate the search in what is believed to be the most promising region. The most promising region is the subregion that is considered the most likely to contain the best solution. Practically, more points are sampled by the promising region (i.e., the union of the subregions generated fixing the $|\Theta_i|$ feasible values of θ_i) than by the surrounding region.

- *Random sampling.* The evaluation of each subregion is achieved by obtaining a random sample of points and evaluating their performance and then using these values to estimate the so called *promising index*, i.e. the index of performance of each the subregion.

- *Estimation of the promising index.* Once a certain number of points have been sampled and evaluated by each partition of the feasible region, then an estimation of the promising index may be achieved as the best performance value found on the points sampled by each partition. A second estimator of the promising index of a subregion may be computed as the average value of the performance of the point sampled by the partition.

- *Backtracking.* Finally, the most promising region is computed as the best promising index among the subregions and the surrounding region. Several backtracking methods have been proposed. An effective backtracking method [84] suggests to backtrack to the whole feasible region if the most promising region is the surrounding region, otherwise the subregion selected as the current most promising region must be partitioned. In this way, the

method can move immediately out of the un-promising region in one transition. This method also requires less computational effort.

Therefore, a key aspect in the design of the NP method is the development of a *partitioning method* (or *partitioning scheme*). The generic partitioning scheme proposed in [57], aims to fix one of the n decision variable $\theta_i \in \theta$ at a time and propose to partition Θ into $|\Theta_i|$ subregions defined by:

$$\sigma_j = \{\theta \in \Theta \,|\, \theta_i = \theta_{ij}\}.$$

where $\theta_{ij} \in \Theta_i$ for $j = 1, \ldots, |\Theta_i|$. Some crucial points are still *i*) the identification of the decision variables $\theta_i \in \theta$, *ii*) the identification of Θ_i[1] or rather of $M_i(\sigma)$ and, finally *iii*) the generation of an intelligent sorting of the variables θ_i.

The first point is strictly connected with the structure of the mathematical model to be optimized, where the decision variables may be different (e.g., continuous, discrete and binary). For what concern the second point, the number of values that each decision variable θ_i can assume is variable. This number depends on the region that the NP is currently partitioning considering the i-th dimension of Θ, i.e. the decision variable θ_i. For instance, in the NP scheme proposed at page 81, the number $|\Theta_i|$ depends on the values fixed for the already evaluated decision variables. Therefore, the first two points are strictly related to the structure of the problem and must be tailored on the specific structure of a solution. Some example of design of a partitioning scheme is shown in Section 2.6 and in Chapter 3.

The third point concerns the decision of the order in which the decision variables should be selected, that is, which variable θ_i should be fixed first, and so forth. Ólafsson and Kim [57] proposed an intelligent partitioning to obtain a ranking of the decision variables $\theta_1, \theta_2, \ldots, \theta_n$. The authors stated that an intelligent partitioning should intuitively increase the probability of correct selection of the most promising region, but it does no assure it in a rigorous manner. Therefore, the decision variables should be sorted randomly or do not sorted at all.

A region constructed using a fixed partitioning scheme is called a valid region given the fixed partition. The set of all valid regions is denoted by Γ. In the NP

[1] In other words, the identification of Θ_i concerns the definition of the function $m(\cdot)$.

method, singleton regions are of special interest, because this particular regions are characterized to include only one point. Therefore, singletons are regions of maximum depth, or rather, regions that cannot be partitioned further. The set of all the singletons is referred as Γ_0, and obviously $\Gamma_0 \subset \Gamma$. The goal of the NP search scheme is to select the region in Γ_0 that has been visited most frequently during the search. Then, this is the proposed optimal solution for the NP method.

The following scheme describes the NP method for a simulation-optimization framework that uses the Rinott's procedure described at page 69 for a two-stage sampling, as proposed in [56].

Initialization Let η be the current iteration of the NP search scheme, then set $\eta = 0$.

Let $\Lambda = \{\lambda_i \,|\, 1 \leqslant \lambda_i \leqslant n, \lambda_i \neq \lambda_j\}$ be a list of indices standing for a ranking of the decision variables $\theta_i, i = 1, \ldots, n$.

Let $\sigma(\eta)$ be the most promising region at iteration η, $s(\sigma(\eta))$ be the surrounding region of $\sigma(\eta)$ (with $s(\Theta) = \emptyset$). As nothing is assumed to be known about location of good solutions before the search is started, then set $\sigma(1) = \Theta$.

We recall that a decision variable θ_i may assume $M_i(\sigma(\eta))$ values. Then, let $\sigma_j(\eta)$ be the j-th subregion of the current most promising region $\sigma(\eta)$, where $\sigma_j(\eta)$ is achieved by assigning at θ_i the j-th value that it can assume. Thus, let $\hat{I}(\sigma_j(\eta))$ be the promising index of the j-th subregion.

Moreover, let $\mathcal{D}_j^i(\eta)$ be the i-th set of N points sampled from the subregion $\sigma_j(\eta)$, where $j = 1, \ldots, M_i(\eta) + 1$ and $1 \leqslant i \leqslant n_1$ in the first sampling stage and $1 \leqslant i \leqslant n_2^j(\eta)$ in the second sampling stage. Let n_1 be the number of sets $\mathcal{D}_j^i$ required in the first sampling stage (with $n_1 \geqslant 2$), while the $n_2^j(\eta)$ is the number of sets $\mathcal{D}_j^i$ for the second stage (this value may be computed using the Rinott's procedure). Eventually, if n_1 has been set-up to 1, then is not required any two stage Ranking & Selection procedure to compute $n_2^j(\eta)$, because this value is naturally set to 0.

Finally, let $\lambda(\eta)$ be the current examined dimension at iteration η,

where $\lambda(\eta) \in \Lambda$ and $\lambda(1) = \lambda_1$.

Step 1 Set $\eta = \eta + 1$.

Given the current most promising region $\sigma(\eta)$, let θ_{λ_η} be the current value depending on $\sigma(\eta)$ will be partitioned, then partition $\sigma(\eta)$ into $M(\eta)$ subregions $\sigma_1(\eta), \ldots, \sigma_j(\eta), \ldots, \sigma_{M(\eta)}(\eta)$, with $M(\eta) = M_{\lambda_\eta}(\sigma(\eta))$ and:

$$\sigma_j(\eta) = \{\theta \in \sigma(\eta) \,|\, \theta_{\lambda_\eta} = \theta_{\lambda_{\eta j}}\}.$$

Then, let $\sigma_{M(\eta)+1}(\eta)$ be the surrounding region of $\sigma(\eta)$, with:

$$\sigma_{M(\eta)+1}(\eta) = \Theta \backslash \sigma(\eta).$$

Goto *Step 2*.

Step 2 For each subregion j, with $j = 1, \ldots, M(\eta) + 1$ and for each i with $1 \leqslant i \leqslant n_1$, then

- Use uniform sampling to obtain a set $\mathcal{D}_j^i(\eta)$ of N sample points from region j.

- For each point $\theta \in \mathcal{D}_j^i(\eta)$, then evaluate $f(\theta)$.
- Estimate the performance of the region from the i-th set $\mathcal{D}_j^i(\eta)$ of points as:

$$X_j^i(\eta) = \min_{\theta \in \mathcal{D}_j^i(\eta)} f(\theta).$$

Goto *Step 3*.

Step 3 For each subregion j, with $j = 1, \ldots, M(\eta) + 1$, calculate the first-stage sample means $X_j(\eta)$ and variance $S_j^2(\eta)$ using the $X_j^i(\eta)$ estimates of the performance of the subregion. Then, use the *Rinott's procedure* to estimate the value of $n_2^j(\eta)$, by selecting a confidence level $1 - \alpha$, an indifference zone parameter $\delta > 0$ and using n_1 as the common first stage sample size.

Goto *Step 4*.

Step 4 For each subregion j, with $j = 1, \ldots, M(\eta)+1$ and $(n_2^j(\eta) - n_1) > 0$ then:

- for each i with $1 \leqslant i \leqslant (n_2^j(\eta) - n_1)$, then
  - Use uniform sampling to obtain a set $\mathcal{D}_j^i(\eta)$ of N sample points from region j.
  - For each point $\theta \in \mathcal{D}_j^i(\eta)$, then evaluate $f(\theta)$.
  - Estimate the performance of the region from the i-th set $\mathcal{D}_j^i(\eta)$ of points as:

$$X_j^i(\eta) = \min_{\theta \in \mathcal{D}_j^i(\eta)} f(\theta).$$

Goto *Step 5*.

Step 5 Let the over all sample mean be the promising index $\hat{I}(\sigma_j(\eta))$ for the subregion j $(j = 1, \ldots, M(\eta) + 1)$, with

$$\hat{I}(\sigma_j(\eta)) = \overline{X}_j(\eta) = \frac{\sum_{i=1}^{n_1} X_j^i(\eta) + \sum_{i=1}^{n_2^j(\eta)} X_j^i(\eta)}{n_1 + n_2^j(\eta)}.$$

Goto *Step 6*.

Step 6 Select the index $\hat{\jmath}_\eta$ of the region with the best promising index, such as:

$$\hat{\jmath}_\eta = \underset{j=1,\ldots,M(\eta)+1}{arg\ min}\ \hat{I}(\sigma_j(\eta)).$$

Goto *Step 7*.

Step 7 If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a subregion of $\sigma(\eta)$, then let this be the most promising region in the next iteration. Otherwise, if the index corresponds to the surrounding region, backtrack to the whole feasible region, which contains the current most promising region (e.g., the surrounding region). In other words, let:

$$\sigma(\eta + 1) = \begin{cases} \sigma_{\hat{\jmath}_\eta}(\eta) & \text{if } \hat{\jmath}_\eta \leqslant M(\eta), \\ \Theta & \text{otherwise.} \end{cases}$$

If $\sigma(\eta+1) = \Theta$, then set $\eta = 1$. Goto *Step 8*.

Step 8 If η is equal to n, that is, all the dimensions of Θ have been fixed then *STOP*: Add $\sigma(\eta+1)$ in the set of singleton regions $\Gamma : 0$ and present $\sigma(\eta+1)$ as the optimum solution; otherwise go back to *Step 1*.

If the partitioning schema has been generated randomly, then we propose the following modification on the preceding schema. Once in *Step 7* the NP method backtracked to the whole feasible region, then produce randomly a new partition schema. The effects of this choice will be discussed in Section 2.6 and in Chapter 3.

Another consideration on the scheme proposed above regards the output of the algorithm. The preceding scheme is used to produce a singleton. Therefore, here we propose a schema for the development of the stopping criterion proposed in [84]. We recall that that stopping criterion aims to select the singleton region that has been visited most frequently. The following procedure implements the previous stopping criterion.

Initialization Let k be the current iteration and let k^{max} be the maximum number of singletons that must be found using the NP method. Let Γ_0 be a set of singleton regions and let $\sigma(k)$ be the singleton found at iteration k.

Step 1 Set $k = k + 1$.
Use the NP method to produce a singleton region σ and set $\sigma(k) = \sigma$. Set $\Gamma_0 = \Gamma_0 \cup \{\sigma\}$. Goto *Step 2*.

Step 2 If k is equal to k^{max}, then goto *Step 3*.
Else goto *Step 1*.

Step 3 Let Γ_0^* be the set of most visited singletons.
For each $\sigma \in \Gamma_0$, compute the number $\upsilon(\sigma)$, such as:

$$\upsilon(\sigma) = \sum_{i=1}^{k^{max}} \zeta(\sigma, i),$$

where:

$$\zeta(\sigma, i) = \begin{cases} 1 & \text{if at iteration } i \text{ results } \sigma(i) \text{ equals to } \sigma, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\Gamma_0^* = \{\hat{\sigma} \,|\, \forall\, \sigma \,:\, \upsilon(\sigma) \leqslant \upsilon(\hat{\sigma})\}$.
Goto *Step 4*.

Step 4 *STOP*: Present $\sigma^* \in \Gamma_0^*$ as the singleton with the best objective function estimate.

2.5 Balanced Explorative and Exploitative Search

In this Section is described a general scheme for designing a search process based on both global and local search called *Balanced Explorative and Exploitative Search with Estimation* (BEESE). This framework has been originally proposed by Prudius and Andradóttir [71] and later more comprehensive developed in an in-depth study by Prudius [70].

The BEESE is a random search based-framework for simulation-based optimization. With exception of the estimation phase, the framework may also be used for deterministic optimization. In this case, the framework is usually referred as BEES.

In this framework, *exploration* refers to searching globally for promising solutions within the entire feasible region Θ, while *exploitation* involves local search of promising subregions of Θ. The *estimation* phase, which is adopted only in a SO context, refers to the process of obtaining a precise function estimates of a solution θ and an improved estimator of the optimal solution. Thus, the search for the optimal solution through the feasible region is pursued by balancing two phases: exploration and exploitation. Prudius remarks the need for maintaining the balance between exploration and exploitation during the search for an optimal solution. In fact, specially for those problems whose structure is unknown, it would be reasonable to start the search by performing a global search stage (i.e., exploring the entire feasible region): in that way is possible to assess how the objective function behaves over the feasible space. This initial stage may help in the identification of a good subregion and

hence the search may be concentrate on the most promising subregion. Once a good subregion is identified, the search must exploit the subregion for better solutions by executing a local search stage (exploitation of the neighborhood). Whenever a subregion has been fully exploited, the search must perform another global search stage aiming to find another promising subregion.

As a consequence of the above discussion, is reasonable to conclude that the effectiveness of the search algorithm depends heavily on the ability of the scheme to identify when it should switch focus from global search (exploration) to local search (exploitation).

As the author suggested, this framework is inspired by other successful frameworks that adopt concepts of exploration and exploitation, such as Tabu Search, Nested Partitions and genetic algorithms. For instance, as we discussed in Section 2.4, the Nested Partitions is based on the diversification and intensification processes that may be considered parallel to the exploration and exploitation processes. In the Tabu Search scheme, diversification refers to the process of identification of a set of elite solutions from the whole Θ, and intensification refers to the evaluation of solutions that are in the neighborhood of the elite solutions.

The estimation of the best solution has been discussed in Section 2.2.1. Our belief is that the most practicable and less computing intensive solution estimator for the BEESE framework consists on the best solution found at a certain algorithm iteration.

Prudius, developed two metaheuristics on the BEES framework: the *Randomized*-BEES(E) or simply RBEES(E) and the *Adaptive*-BEES(E) or ABEES(E) for short. For both development he demonstrated their almost sure convergence. The RBEES and ABEES are two random search methods that proved to have similar performance [70].

Both scheme need the definition of two context specific schemes for the sampling of solution during the exploration and exploitation phase. With respect to the exploration phase, the global search. Recalling that exploration and exploitation are respectively the global and local search of promising solutions, the schemes for sampling solutions are called *global sampling procedure* $\mathcal{G}$ and *local sampling procedure* $\mathcal{L}(\theta_{\eta-1})$. The global sampling procedure $\mathcal{G}$ samples a new solution θ from

the whole Θ; the local sampling procedure $\mathcal{L}(\theta_{\eta-1})$, samples at the iteration η a new solution $\theta \in \Theta$ that is neighbor to $\theta_{\eta-1}$.

In both schemes, at the η-th iteration: θ is the solution currently examined, i.e., is a candidate solution generated using the procedure $\mathcal{G}$ or $\mathcal{L}(\theta_{\eta-1})$; θ_η is the best solution found yet; θ^* is the best solution found at the conclusion of the search process.

The effectiveness of the search scheme strictly depends on the choice of the sampling procedures $\mathcal{G}$ and $\mathcal{L}$.

Thus, in the following we provide a description of the RBEES and ABEES meta-heuristics.

2.5.1 RBEES

The RBEES is a random search method designed to balance global and local search. The RBEES framework, at any iteration, with probability $0 > p \leqslant 1$ the sampling procedure $\mathcal{G}$ is used, and with probability $1 - p$ the sampling procedure $\mathcal{L}$ is used. This creates a balance in the use of exploration and exploitation during all stages of the search. An user must properly define the value of p in order to find the right balance between the exploration and estimation process. For instance, if $p = 1$ the RBEES is used as a pure random search method, i.e., only global search is performed.

The following is the RBEES search scheme with a few of modification in order to let the algorithm be consistent with the general scheme shown in Figure 2.1.

Initialization Let η be the current iteration of the RBEES search scheme, then set $\eta = 0$.
If an initial solution has been set-up, then let θ be the the initial solution. Otherwise, sample a solution θ using the global sampling procedure $\mathcal{G}$.
Evaluate $f(\theta)$ and set $\theta_\eta = \theta$.

Step 1 Set $\eta = \eta + 1$.
Get a variate u from a uniform random variable $U\{0, 1\}$.

if $u \leqslant p$, then sample a solution θ using $\mathcal{G}$
else sample a solution θ using $\mathcal{L}(\theta_{\eta-1})$.
Evaluate $f(\theta)$ and goto *Step 2*.

Step 2 If $f(\theta) \geqslant f(\theta_{\eta-1})$ and the goal is to maximize or $f(\theta) \leqslant f(\theta_{\eta-1})$ and the goal is to minimize, then set $\theta_\eta = \theta$.
Goto *Step 3*.

Step 3 If stopping criterion is not satisfied, then goto *Step 1*
else goto *Step 4*.

Step 4 *STOP*: Present $\theta^* = \theta_\eta$.

The previous algorithm, every time the local sampling procedure $\mathcal{L}$ is chosen for sampling the current candidate solution θ at a specific iteration η, it focuses the search around the solution that proved to have the highest objective function values, i.e., θ_η.

Classical stopping criterion can be used. Thus, the search process can be stopped *i*) after that a fixed number of iterations are performed, *ii*) if a time limit is reached, and *iii*) if the current optimal solution is within few units from a good upper-bound (lower-bound) in a maximization (minimization) process. Definitely, combination of the previous stopping criterion are desirable.

An intelligent stopping criterion can be defined reasoning on the alternation of the exploration and exploitation phases. Reasonably, the need for exploration may decrease with the rise of the iterations. After a sufficiently large number of iterations, the whole feasible region is generally enough exploited and the sub-region that includes the global optimal solution is identified. In this case, is possible to reduce the effort due to the exploration by operating on the value of the probability p. In fact, p can be obtained as result of a function that depend on some meaningful data, e.g., *i*) the iteration number η, and *ii*) the sample path of the method up to iteration $\eta - 1$. This idea may recall to the cooling process in the Simulated Annealing metaheuristics [33].

2.5.2 ABEES

The ABEES strategy adaptively alternates between (local) sampling from the neighborhood of a current solution and (global) sampling in the entire space of feasible solutions. As stated before, the adopted sampling technique is a key performance factor of the methodology.

Contrarily at the RBEES that switch at each iteration from global to local search with a probability $1 - p$ (and *vice versa*), the ABEES search strategy aims to iteratively change the search focus (exploitation or exploitation) in an adaptive way. So, the ABEES reviews the search nature every k iterations in order to evaluate if the current search process is achieving good results or not. An advanced approach require to perform k_g iterations with the global search before to review the search nature, and analogously k_l iterations in the local search stage. Empirical studies proved that generally is better to set $k_l > k_g$. This is reasonable is we consider that the exploitation of a promising neighborhood is a key factor in the updating of the search nature. In fact, the ABEES framework switch from local search to global search every time a neighborhood appear to be fully exploited.

Reminding that θ_η is the best solution found at the iteration η, let also θ^l be the best point found the last time local search was performed (note that its function value may be worst than the previous best point found during the previous local search stage), and θ^k be the point corresponding to the best solution found at the last review. Moreover, the corresponding function values are $v_\eta = f(\theta_\eta)$, $v^l = f(\theta^l)$ and $v^k = f(\theta^k)$.

Let Δ be the improvement in the function value between the current and preceding reviews and D the distance between the points where the corresponding function values were achieved. The ABEES also requires two user-defined thresholds, namely the distance threshold d and the improvement threshold δ.

Initialization Let η be the current iteration of the ABEES search scheme, then set $\eta = 0$. Let *counter* be the number of iterations performed during the current search nature, then set $counter = 0$. Let LS be a boolean flag that is *true* if the search nature is local search, $false$ otherwise, then set $LS = false$.

If an initial solution has been set-up, then let θ be the the initial solution. Otherwise, sample a solution θ using the global sampling procedure $\mathcal{G}$.
Evaluate $f(\theta)$ and set $\theta_\eta, \theta^k, \theta^l = \theta$. Set $v_\eta, v^k, v^l = f(\theta)$.

Step 1 Set $\eta = \eta + 1$ and $counter = counter + 1$.
If LS is $false$, then sample a solution θ using $\mathcal{G}$
else sample a solution θ using $\mathcal{L}(\theta_{\eta-1})$.
Evaluate $f(\theta)$ and goto *Step 2*.

Step 2 If $f(\theta) \geqslant f(\theta_{\eta-1})$ and the goal is to maximize or $f(\theta) \leqslant f(\theta_{\eta-1})$ and the goal is to minimize,
then set $\theta_\eta = \theta$ and $v_\eta = f(\theta)$.
Goto *Step 3*.

Step 3 If LS is $false$ and $counter = k_g$ or LS is $true$ and $counter = k_l$, then

- If the goal is to maximize, then set $\Delta = (v^k - v_\eta)/v^k$.
- Else if the goal is to minimize, then set $\Delta = (v_\eta - v^k)/v_\eta$.
- Compute D between θ_η and θ^k using such distance measure.
- Set $\theta^k = \theta_\eta$ and $v^k = v_\eta$ and $counter = 0$.

Goto *Step 4*.

Step 4 If LS is $true$ and $\Delta \leqslant \delta$, then set $LS = false$, $theta^l = theta_\eta$ and $v^l = v_\eta$.
Otherwise, if LS is $false$ and $\Delta \leqslant \delta$ then

- If the goal is to maximize and $(v^l - v_\eta)/v^l \geqslant \delta$, or the goal is to minimize and $(v_\eta - v^l)/v_\eta \geqslant \delta$,
  then set $LS = true$.

Else if LS is $false$ and $D \leqslant d$, then set $LS = true$.
Goto *Step 5*.

Step 5 If stopping criterion is not satisfied, then goto *Step 1*
else goto *Step 6*.

Step 6 *STOP*: Present $\theta^* = \theta_\eta$.

The ABEES algorithm can switch every k_g iterations from global to local search in two ways: *i*) whenever the improvement Δ is small (less than or equal to the user-defined threshold δ), but the method finds a substantial improvement in the objective function value compared to the improvement found during the last time local search has been performed (in this case, a promising sub-region has been identified); *ii*) when the improvement Δ is small, but the distance D between successive reviews is small (less than or equal to the user-defined measure d). In both cases, the local search flag LS is set to $true$. Vice versa, the algorithm switches from local to global search if no meaningful improvement has been achieved during the last k_l algorithm iterations (in this case, LS is set to $false$).

Like fore the RBEES, the ABEES needs the definition of both the global and local sampling procedures. Nevertheless, the ABEES scheme also requires a distance measure. In literature, there are several distance measures that have been defined for different classes of problems (an example is shown in Chapter 3 on a scheduling problem).

As discussed in Section 2.5.1 at page 88, classical stopping criterion can be used also for the ABEES metaheuristic scheme.

Contrarily to the RBEES, the ABEES has natively an adaptive search scheme. Therefore, we propose an advanced stopping criterion that regard the number of review executed consecutively without switching on local search. In fact, if the ABEES execute consecutively $r_g = g(k_g)$ reviews without switching on the exploitation phase, then all the sub-regions have been fully exploited and the search may be stopped. The function g depends on the number of iterations k_g executed during each exploration phase before each review of the nature search.

2.6 Simulation-based Optimization of a Manufacturing System

In this Section is proposed an example of optimization via simulation of a *multi-product flow-shop* manufacturing system that operates in a supply chain. In particular, here is discussed a common problem in supply chain management, i.e., the

allocation of discrete resources. However, in a stochastic environment, the allocation of discrete resources appears as a problem very difficult to solve. Therefore, we are interested to show the potentiality of the SO approach to a typical *Resource Allocation Problem* (RAP). SO is applied to the problem described below using the NP method and the RBEES and ABEES algorithms.

The manufacturing system is depicted in Figure 2.4 as a queuing network model with two-classes of products ($C1$ and $C2$) and $m = 10$ buffered machines, with a FIFO queue policy. The model has been already proposed in [82].

The classes of products have different arrival distributions. In particular the class $C1$ has an exponential arrival distribution with a rate λ, while the class $C2$ has a hyper-exponential arrival distribution with parameters λ_1, λ_2 and α. Both classes arrive at any of the machines 1–4 (i.e. with the same probability 0.25), and leave the system after have been processed by three different processing. Part routing within the queuing network is class dependent. As shown in Figure 2.4, class $C1$ leaves the system by machine 9, while class $C2$ leaves by machine 10.

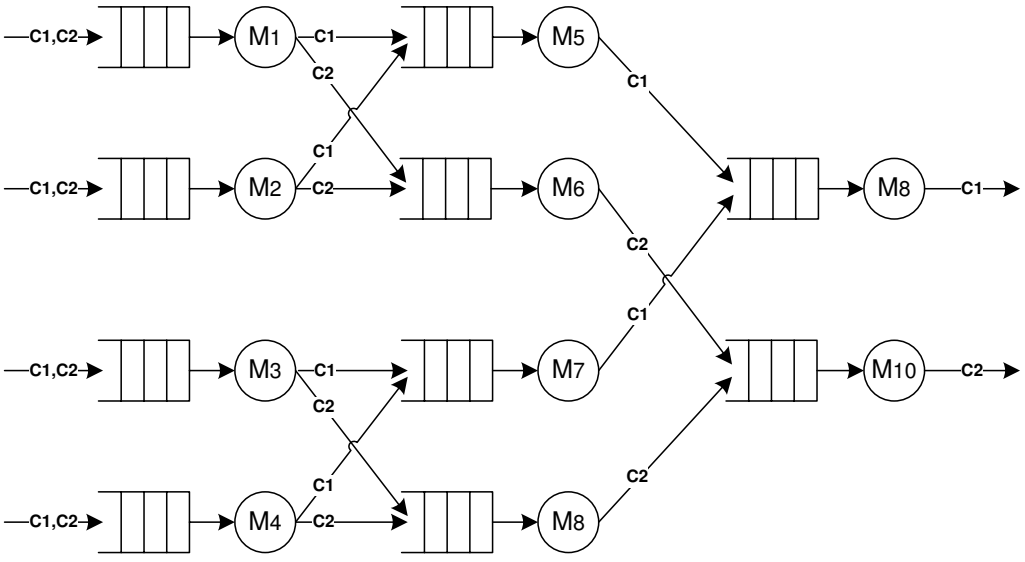


Figure 2.4: Queuing network model for a manufacturing system.

The processing time of all the machines is exponentially distributed, and in particular, μ_i is the processing rate for the machine i, with $i = 1, \ldots, m$.

Let B_i be the buffer size for the queue related with the machine M_i, with $i = 1, \ldots, m$ and B_i finite. Then, the finiteness of the buffer size for each of the m queue of the production line makes possible the definition of an interesting and simple-to-understand resource allocation problem. In particular, the goal is in distributing optimally buffer spaces to different queues given a limited budget $b^{max}$for the buffer sizes. In real systems, the finiteness of the buffer sizes may affect the performance of a manufacturing system by producing *starvation* and *blocking* phenomena, and therefore reducing system throughput, resources utilization and increasing tremendously the cycle time. Here, we consider the problem of allocating 12 buffer units, among the 10 different buffers. Let T be the throughput of the production line, then the SO problem that we are interested to solve is the following:

$$maximize \quad f(B) = E[T(B)] \tag{2.3a}$$

subject to

$$\sum_{i=1}^{m} B_i = b^{max} \tag{2.3b}$$

$$B_i \geqslant 0 \qquad \forall\, i = 1, \ldots, m \tag{2.3c}$$

$$T \geqslant 0, \in \mathbb{R} \tag{2.3d}$$

where $B = (B_1, B_2, \ldots, B_m)$ is a solution of the problem and the objective function $f(B)$ is an estimate of the random variable $T(B)$. With a fixed set of parameters (i.e., arrival times and service times), only the decisional variables B_i can affect the performance of the system throughput. For the previous system, the throughput must be computed considering the number of products $C1$ and $C2$ that leave the production line in a fixed time period (that generally do not include the transient period).

In this example, we set-up the parameter b^{max} as 12. Note that the total number of point in the feasible region is quite high: in particular, there are 293,930 different solutions B which satisfy the constrain in (2.3b). While the simulation time for each

system configuration is not very long, the total simulation time for all the designs is not affordable. Therefore, an heuristic approach is required.

Moreover, deterministic optimization cannot provide any solution to the mathematical model (2.3), notwithstanding the structure is quite simple. In fact, the evaluation of the objective function (2.3a) requires the evaluation of the performance of a system which operates in a stochastic-dynamic environment. Therefore, only discrete-event simulation is well capable of representing the system described in Figure 2.4 and to evaluate the system throughput for each feasible solution.

To perform SO using the BEES and NP frameworks, some details must be addressed to the generation of a feasible resource assignment. In particular, we propose a brief description about the implementation of two possible procedures for sampling a solution globally and locally within the BEES framework. Successively, an explanation about the use of the NP method on the above SO problem is also provided.

Solution generation In the BEES framework, both RBEES and ABEES metaheuristics use a global and a local sampling procedure. A possible global sampling procedure for this this simple SO problem may be defined using a uniform random variable U within the range $[1, m]$, and iteratively producing a variate u such as the value of the buffer B_u is increased by one until the summation over all the B_i satisfy the constraint (2.3b), with $i = 1, \ldots, m$.

Then, given a feasible assignment $B^1 = (B_1, B_2, \ldots, B_m)$, a local sampling procedure can be designed by generating a neighbor assignment B^2. Therefore, a key aspect is the definition of a distance measure able to specify if two assignment are in the same neighborhood or not. A possible distance measure is the following.

$$d(B^1, B^2) = \sum_{i=1}^{m} \left| B_i^1 - B_i^2 \right| \tag{2.4}$$

Thus, a local sampling procedure may be defined as a procedure that, using a uniform random variable U defined within the range $[1, m]$, iteratively generates the variates u_1 and u_2 until u_1 is not different by u_2 and B_{u_1} or B_{u_2} is greater than 0. Once two buffer with the desired capacity have been found, the procedure increase by one a buffer chosen randomly between B_{u_1} or B_{u_2} and decrease the

related buffer size. Obviously, if one of the two buffer has capacity equal to 0, then this buffer size is increased and the other buffer capacity is decreased. The capacity are increased/decreased by one in order to produce two assignment that are within a maximum distance of 2, according to the distance measure in (2.4).

The NP method requires the specification of such method for the evaluation of the number $M_i(\sigma)$ of possible values that a decision variable B_i, with $i = 1, \ldots, m$ can assume within the current most promising region σ and, therefore, a procedure for the sampling of points from i) every subregion of σ, and ii) the related surrounding region.

First and foremost, an explanation about the nature of a generic region σ is required. A region is banally a partial assignment of the m buffer capacities. Therefore, for the current most promising region σ found at iteration η, let b^η be the budget for buffer sizes already allocated at iteration η and n_0^η be the number of buffers whose size has been set to zero in region σ at iteration η. Then, the number of possible values $M_i(\sigma)$ that can be assigned to the i-th buffer B_i is exactly $b^{max} - b^\eta + 1$. The plus one value takes into account the possibility to assign the 0 value to the buffer size B_i.

In a simple way, the sampling of a point from a region σ at the iteration η is achieved in at most $b^{max} - b^\eta$ steps, i.e. the number of steps required to allocate all the residual buffer spaces. Therefore, let $\mathcal{B}$ be the set of all the un-allocated buffer spaces and $b^r = b^{max} - b^\eta$ be the buffer spaces not yet allocated. Then, let U be a uniform random variable defined within the range $[1, b^r]$, then chosen at random a buffer $B_i \in \mathcal{B}$ and generated a variate u from U, set $B_i = B_i + u$ and $b^r = b^r - u$, until $b^r > 0$. If some $B_i \in \mathcal{B}$ have never been selected, then set $B_i = 0$.

Points from the surrounding region $s(\sigma)$ (if the surrounding region is not null) are sampled likewise. In particular, all the b^{max} buffer spaces are assigned to all the buffers as described above (i.e., $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$). During the buffer sizes allocation, for each B_i already fixed in σ, the allocation for B_i in $s(\sigma)$ must be different.

Numerical experiments Here is reported a numerical experiment for the queuing network model depicted in Section 2.6. For the following experiment, we fixed

the subsequent parameters: $\lambda = 12\,[\text{products/min}]$, $\lambda_1 = 10$ *[products/min]*, $\lambda_2 = 30$ *[products/min]* and $\alpha = 0.9$, $\mu_1 = \mu_2 = 4$ *[products/min]*, $\mu_3 = \mu_4 = 3$ *[products/min]*, $\mu_5 = \mu_6 = \mu_7 = \mu_8 = 4$ *[products/min]*, $\mu_9 = \mu_{10} = 5$ *[products/min]*. The simulated time is set up to 8 hours.

For the RBEES we tested the metaheuristics with the probability p equals to 0.3, 0.5 and 0.8. The ABEES has been tested with $d = 4$, $\delta = 0.01$ and the following couple of values for k_g and k_l: $(5, 50)$, $(50, 5)$, $(25, 25)$

The previous is a simple and replicable experiment that shows how contextualizing a SO algorithm on a realistic system.

Some other interesting and more complicated examples of optimization through simulation in production logistics have been proposed by the authors in [46, 48]. In particular, in [46] a just in time production line with the celebrated Kanban-based mechanism of production control, and in [48] a *Flexible Manufacturing System* (FMS), have been optimized using an open-source discrete-event simulator with some metaheuristics for SO suitably integrated into the simulator.

2.7 Conclusions

In this Chapter has been introduced *simulation-based optimization*. A generic schema for the design of a simulation-optimization procedure has been proposed. Therefore, has been discussed some procedure for the correct selection of a system configuration in a simulation-optimization procedure. Therefore, commercial optimization packages for discrete-event simulators are discussed, with particular attention to the need of solving complex mathematical models. Therefore, two recently developed metaheuristics generally applied to a simulation-optimization environment have been discussed. Finally, an example of tuning of the introduced metaheuristics on a resource allocation problem related to a production line is discussed.

Chapter 3

Simulation-based Optimization Techniques for the Quay Crane Scheduling Problem

Maritime terminals of pure transhipment are emerging logistic realities in long-distance containerized trade. Here, complex activities of resource allocation and scheduling should be optimized in a dynamic, non deterministic environment. The assignment of expensive quay cranes to multiple vessel-holds for container discharging and loading operations is a major problem, whose solution affects the operational performance of the whole terminal container. In OR literature, this problem is known as the quay crane scheduling problem. With the objective of minimizing the vessel's overall completion time, we first give our IP formulation and then, under the more realistic assumption that discharge-loading times are non deterministic, we focus on a simulation-based optimization approach which embodies the IP formulation. Two different simulation optimization algorithms are tailored to the problem: *Balanced Explorative and Exploitative Search* and *Nested Partitions*. Numerical results are also presented on real vessel data and on randomly generated instances.

3.1 Introduction

The world container fleet amounts to about 23.2 million TEUs (twenty-foot equivalent units) and in 2006 the container throughput reached 440 million TEUs [90]. Containerized trade is forecasted to grow by an average annual rate of 5.32% until the year 2025 [89]. As a result of this trend, the number of maritime and inland container terminals worldwide keeps increasing. Competition has become both price driven and service driven and, therefore, the success of an individual company will depend on its ability to fulfill customer demand with high standard quality service, while keeping operations lean.

Maritime container terminals are the most important crossroads for transshipment and intermodal container transfers, based on the spokes-hub distribution paradigm. These facilities have different layouts and they are typically composed by heterogeneous sets of resources deployed within each port sub-area. According to Steenken et al. [87], the main sub-areas are *i*) the ship operation area (i.e., the *quay*), *ii*) the import/export stacking area (i.e., the *yard*) and *iii*) the truck and train operation area. Referring to the operations that occur within the quay and yard areas, the most common resources are cranes and shuttle vehicles. Quay cranes (QCs) are usually of two types: *rail-mounted gantry cranes* (RMGCs) and *rubber-tired gantry cranes* (RTGCs)[1]. Shuttle vehicles are selected according to how container transfer occurs from the quay to the yard and vice versa: most European and North-American container terminals are generally based upon the "Direct

[1] Another feature that characterize the QCs is the number of trolleys. The mojority of the QCs are of *single-trolley* type, while few terminals are equipped with *dual-trolley* QCs. Single-trolley QCs move the containers from the ship to the shore either putting them on the quay or on a vehicle (and *vice versa* for the loading cycle). Dual-trolley QCs have a main trolley that moves the container from the ship to a platform while a second trolley picks up the container from the platform and moves it to the shore (and *vice versa* for the loading cycle). Contrarily to single-trolley cranes, that are man-driven, in dual-trolley cranes only the first trolley is man-driven, while the second is automatic. Thus, dual-trolley QCs are characterized by higher performances. The performance in operations of QCs is in the range of 22–30 $boxes/h$.

Transfer System" (DTS), which implies the use of *straddle carriers*, special vehicles able to pick-up/set-down and transfer one or more containers per time.

Stahlbock and Voß [86] claim that container handling (i.e., stacking and transport operations) is a key factor for a container terminal's efficiency. In this context, a complex scheduling problem, which arises when multiple quay cranes are assigned to the same ship with the aim of performing discharge and loading operations, is known as the *quay crane scheduling problem* (QCSP).

The goal of the QCSP is to planning the quay crane movements to load or unload ships considering a known *stowage plan* and under a specific goal, e.g. the minimization the overall completion time (makespan minimization). The QCSP is a particular *m-parallel machines scheduling problem* [10, 68, 69], where quay cranes are the machines and a task is defined as the discharge or loading of all containers related to the *deck* or the *hold* of a specific vessel *bay*.

Consequently, precedence relationship must be considered between tasks related to the same bay: in particular, during discharging operations, tasks stowed on the deck must be performed before tasks in the hold of the same ship-bay; also, the loading operation in a hold must precede the loading operation on the deck of the same bay. Moreover, discharging operations must precede loading operations on the same bay. To avoid collisions for QCs working on adjacent bays, non-simultaneity constraints between tasks must also taken into account. Generally, two adjacent QCs must be apart from each other by approximately one bay so that they can simultaneously perform their tasks without interference.

Some other considerations are referred to the QC types. RTGs are more flexible in operation, while RMGs are more stable. Therefore, if the QCSP refers to RMGs, i.e. to QCs that travel on the same track, then non-crossing constraints must be taken into account. This is not necessary for RTGs, but we must consider that the flexibility of RTGs has a cost in terms of time (that is the time required to move from a bay to another bay by crossing other QCs).

The solution of the QCSP has been successfully dealt with in literature by using both deterministic approaches (and solving the relaxation of the Integer Programming formulation) and metaheuristics algorithms [19, 32, 49, 76].

In real life management of logistics at a maritime container terminal, the QCSP

arises as a decisional step within the discharge/loading process; thus, we address the issue of using the solution of the QCSP within a simulation model of the above process. The simulation model has to evaluate the key performance measure that should be optimized. Here we show how a simulation-based optimization is a cost-effective technique in terms of results realism and quality of the solution returned.

The remainder of this chapter is organized as follows. In the next section, we provide a detailed description of the logistic processes set around the discharge/loading operations in a maritime container terminal. Afterward, we propose a mathematical formulation of the QCSP based on a positional notation. In the following section, we describe two simulation-based optimization approaches to the QCSP using the *Balanced Explorative and Exploitative Search* (BEES) and the *Nested Partitions* (NP) frameworks that we already introduced in Chapter 2. Numerical results using both real vessel data and randomly generated instances are provided and compared with the deterministic problem solution obtained through the CPLEX solver. Conclusions are reported in the last section.

A preliminary version of this Chapter appeared in Legato, Mazza and Trunfio [44].

3.2 Problem Description

In the Chapter 1 we have described the whole processes that arise in a maritime container terminal. In particular, we focused our attention onto the port of Gioia Tauro. Moreover, we have pointed the attention on the importance of resource assignments (i.e., berth slots and QCs) for the vessel arrival-service-departure process and the consequent construction of a suitable weekly plan for the berth allocation office. Successively, we described both vessel discharge/loading and container transfer processes of the port using some modelling paradigms (e.g., *Event Graphs*, *Petri Nets*, etc.). The main focus was on the representational capabilities offered by some modelling languages used to incorporate both the low level operational policies and work rules of the above process and the specific scheduling constraints involved in the assignment of QCs to the set of tasks.

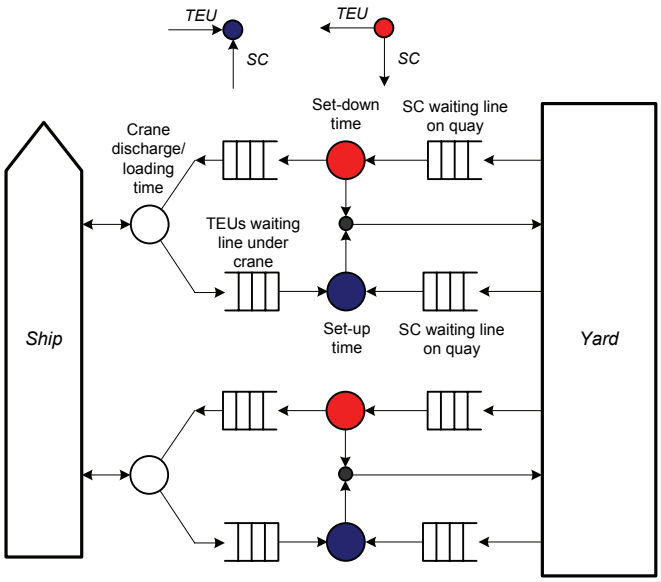


Figure 3.1: The vessel discharge/loading and container transfer processes

On the other hand, to further improve the efficiency of berth operations, a very important role is also played by the QCs and their ability to perform container discharge/loading operations. As many as 6 units of this expensive handling equipment can be deployed to serve the latest generation containerships during an operational work-cycle.

A QC operates in the berth area by moving (on wheel or rail) in horizontal directions to reach different bays within the same vessel or on different vessels. For both discharge and loading operations, a very restricted area (e.g. a 6-slot space) for buffering a limited number of containers is naturally provided at the basis of each QC.

As we anticipated in Chapter 1, for those maritime container terminal that relies on the *indirect transfer system*, a QC performs discharge and loading operations as follows. When performing discharge operations, a quay crane picks-up containers from the vessel and "feeds" them to straddle carries (SCs) which provide for their transfer from the quay area to the assigned yard positions within the terminal storage area. As one may observe in Figure 3.1, the discharge process from the ship to the yard features a joining point (in blue) between the unloaded container and the

SC sent for its pick-up and transfer to the yard. As far as loading operations are concerned, a QC picks-up containers delivered from the terminal yard by the SCs and places them on the ship in the assigned vessel holds. Figure 3.1 accounts for this process from the yard to the ship as well: in particular, the forking point (in red) represents the physical separation carried out by an SC when it first sets-down the container in the quay crane buffer area and then returns (empty) to the yard to retrieve other containers.

From here on, we concentrate on the QCSP. The objective of our study is to determine the crane schedule (in other words, which and in what order tasks should be assigned to the single QCs) to minimize the vessel's overall completion time. For that goal, we take into account that:

- a minimum distance is left between quay cranes to avoid boom collision (i.e. non-simultaneity constraints);
- some holds must be operated before others (precedence constraints);
- not every crane is available immediately (release constraints).
- each crane is of RTG type.

Other requirements may be necessary, e.g. when a QC that is already assigned to the vessel is re-assigned to another vessel at a specific time.

For problem solution, in the following sections we propose both a mathematical model and two simulation-based optimization approaches.

3.3 Mathematical Formulations

This section includes a Mixed Integer Programming (MIP) mathematical formulation for the QCSP derived from the Kim and Park [32] model, and also proposes a new MIP formulation of the QCSP based on a positional notation.

The scheduling of the operations of the QCs must be identified under the constraints shown below: *i*) Each QC can operate after its earliest available time (in fact, a QC may be available only after it completes a previously assigned work on

another ship or when the assigned crane operator starts its workshift); *ii*) Some tasks must be performed before others (e.g., whenever two holds are in the same bay); *iii*) There are some tasks that cannot be performed simultaneously (e.g., neighboring holds cannot be performed at the same time to avoid crane collisions). The following notations are used for a mathematical formulation.

We refer to mathematical models for RTGs, therefore we do not provide non-crossing constraints among cranes for both mathematical formulations.

3.3.1 Kim and Park Formulation for RTG Quay Cranes

In this section is shown the model proposed in [32] and successively reviewed and strengthened in [54]. In particular, we focus our attention on problems with QCs that are not on the same track and thus they can cross each other (i.e. RTGs).

The following notations are used for the mathematical formulation.

Let $\Omega = \{1, \ldots, n\}$ be a set of n tasks and $K = \{1, \ldots, q\}$ be a set of q quay cranes. Each task amounts to perform a fixed number of container moves (discharge/loading) which require a non deterministic number of time slots to be carried out. As first approximation, one may formulate a MIP model by resorting to the use of the average values for the above processing times. Thus, let p_i be the processing time of task $i \in \Omega$.

The set $\Psi = \{(i, j)|i, j \in \Omega\}$ of pairs of task describes non-simultaneity relationships between task pairs, i.e. for each $(i, j) \in \Psi$ task i must be completed before task j starts or task i must start before task j is completed. Moreover, the set $\Phi = \{(i, j)|i, j \in \Omega\}$ of ordered pairs of tasks stands for the precedence relationships between task pairs, i.e. for each $(i, j) \in \Phi$ task i must be completed before task j starts. Note that $\Phi \subseteq \Psi$.

Let r_k be the release time of quay crane $k \in K$, with and note that, once again, this is considered as a deterministic value.

In the end, let t_{ij} be the time required by a QC to move from the bay related to the task i to the bay of the task j, with $i, j \in \Omega$. Obviously, tasks from the same bay have $t_{ij} = 0$. Let us also introduce t^k_{0j} and t^k_{iT}, respectively as the time required by the QC $k \in K$ to move *i*) from its initial position to the task $j \in \Omega$, and *ii*)

from the position of the task $i \in \Omega$ to its final position. Thus, in this notation, the start position and the final position of a QC are indicated as 0 and T. Therefore, for notational convenience define $\Omega^0 = \Omega \cup \{0\}$ and $\Omega^T = \Omega \cup \{T\}$.

M is a suitably big number (note that the value of M generally affects the performance of the search process using commercial solver for MIP models).

The problem is modeled on a graph $G = (V, A)$, where $V = \Omega \cup \{0, T\}$ and $A \subseteq V \times V$.

The formulation uses the following variables.

- X_{ij}^k, for each $(i, j) \in A$ is equal to 1 if and only if task i is performed by crane k immediately before task j, 0 otherwise. The task j is the first task of QC k if $x_{0j}^k = 1$, and similarly, the task i is the last task of QC k if $X_{iT}^k = 0$.
- Z_{ij}, for each $(i, j) \in A$ is equal to 1 if task i is completed before task j starts, 0 otherwise (obviously, if $Z_{ij} = 1$ therefore $Z_{ji} = 0$ and *vice versa*).
- D_i, for each $i \in \Omega$ is the completion time of i.
- C^k, for each QC $k \in K$ stands for the completion time of k.
- W, is the overall completion time of the vessel, i.e. is the makespan.

The Kim and Park QCSP without non-crossing constraints can be formulated as follows.

$$minimize \quad \alpha 1 W + \alpha 2 \sum_{k \in K} C^k \tag{3.1a}$$

subject to

$$C^k \leqslant W \qquad \forall\, k \in K \tag{3.1b}$$

$$\sum_{j\in\Omega} X^k_{0j} = 1 \qquad \forall\, k \in K \tag{3.1c}$$

$$\sum_{i\in\Omega} X^k_{iT} = 1 \qquad \forall\, k \in K \tag{3.1d}$$

$$\sum_{k\in K}\sum_{i\in\Omega^0} X^k_{ij} = 1 \qquad \forall\, j \in \Omega \tag{3.1e}$$

$$\sum_{j\in\Omega^T} X^k_{ij} - \sum_{j\in\Omega^0} X^k_{ji} = 0 \qquad \forall\, i \in \Omega, \forall\, k \in K \tag{3.1f}$$

$$D_i + t_{ij} + p_j - D_j \leqslant M(1 - X^k_{ij}) \qquad \forall\, i, j \in \Omega, \forall\, k \in K \tag{3.1g}$$

$$D_j - p_j - D_i \leqslant MZ_{ij} \qquad \forall\, i, j \in \Omega \tag{3.1h}$$

$$Z_{ij} = 1 \qquad \forall\, (i, j) \in \Phi \tag{3.1i}$$

$$Z_{ji} = 0 \qquad \forall\, (i, j) \in \Phi \tag{3.1j}$$

$$Z_{ij} + Z_{ji} = 1 \qquad \forall\, (i, j) \in \Psi \setminus \Phi \tag{3.1k}$$

$$r_k - D_i + t^k_{0i} + p_i \leqslant M(1 - X^k_{0i}) \qquad \forall\, i \in \Omega, \forall\, k \in K \tag{3.1l}$$

$$D_j + t^k_{jT} - C^k \leqslant M(1 - X^k_{jT}) \qquad \forall\, j \in \Omega, \forall\, k \in K \tag{3.1m}$$

$$X^k_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in A, \forall\, k \in K \tag{3.1n}$$

$$Z_{ij} \in \{0, 1\} \qquad \forall\, i, j \in \Omega \tag{3.1o}$$

$$D_i, C^k \geqslant 0 \qquad \forall\, i \in \Omega, \forall\, k \in K \tag{3.1p}$$

We now explain the model (3.1). The multi-objective function (3.1a) is a linear combination of the makespan, defined in constraints (3.1b), and the sum of the crane completion times computed in constraints (3.1m). In the previous, minimizing the makespan coincides with the minimization of the overall completion time of the vessel, while minimizing the completion time of the cranes maximizes their productivity. Thus, Kim and Park [32] assumed that $\alpha 1 \gg \alpha 2$, since minimizing the makespan is considered a primary objective. Therefore, $\alpha 2$ is usually set to 0.

Constraints (3.1c) and (3.1d) respectively select the first and last tasks for each QC. Constraints (3.1e) specify that every task can only be assigned to one crane[2].

[2]The original constraints were for all $i \in \Omega$, but obviously this is was a typing error, and

Constraints (3.1f) guarantee that on each QC tasks are performed in a well-defined sequence. Constraints (3.1g) simultaneously determines the completion time for each task and eliminates sub-tours. Constraints (3.1h) have been introduced in [54] to make stronger the previous constraints. Constraints (3.1i) and (3.1j) guarantee that task i will be processed before task j if there is a precedence relationship between them. Note that also this constraints have been reviewed in [54]. Constraints (3.1k) ensure that tasks i and j are not processed simultaneously if, as a pair, they belong to the non-simultaneity set. Constraints (3.1l) and (3.1m) respectively guarantee that a task cannot be assigned to a crane before the crane has been released, and that the last task performed by each crane must be completed within the crane completion time. Finally, constraints (3.1n)–(3.1p) are the constraints on the decision variables.

Note that this model has $O(|\Omega|^2 \cdot |K|)$ variables and $O(|\Omega|^2 \cdot |K|)$ constraints.

3.3.2 A Positional Formulation of the QCSP

We now introduce a new formulation of the QCSP that may be addressed to maritime container terminals equipped with RTG QCs. The formulation assume that each QC has a schedule of tasks that must be performed in a fixed sequence, so the goal of the model is to compute the position in which each task will be performed on a each QC.

The notations of the following MIP formulation is taken by the model defined in section 3.3.1, with the following modifications.

Let $P = \{1, \ldots, m\}$ be a set of m positions. Considering that in the worst case all the tasks must be performed by the same QC (therefore all the positions of the QC will be allocated to the tasks), we will set $|P| \equiv |\Omega| = n$. For notation convenience, we also define the set $P^- = P \backslash \{m\}$.

In the following, we re-define the $\overline{X}$ and $\overline{Z}$ decisional variables:

- X^k_{ih}, for each $i \in \Omega$ is equal to 1 if and only if task i is performed in position h by crane k, 0 otherwise.

therefore we modified the constraints by letting the summation to vary for all $i \in \Omega^0$.

- Z_{ij}, for each $(i,j) \in \Psi$ is equal to 1 if task i is completed before task j starts, 0 if i starts after the completion of task j (obviously, if $Z_{ij} = 1$ therefore $Z_{ji} = 0$ and *vice versa*);

Our formulation of the QCSP without non-crossing constraints is formulated as follows.

$$minimize \quad \alpha 1 W + \alpha 2 \sum_{k \in K} C^k \tag{3.2a}$$

subject to

$$C^k \leqslant W \quad \forall\, k \in K \tag{3.2b}$$

$$\sum_{k \in K} \sum_{h \in P} X^k_{ih} = 1 \quad \forall\, i \in \Omega \tag{3.2c}$$

$$\sum_{i \in \Omega} X^k_{ih} \leqslant 1 \quad \forall\, h \in P, \forall\, k \in K \tag{3.2d}$$

$$\sum_{\substack{j \in \Omega \\ j \neq i}} X^k_{jh} \geqslant X^k_{i\,h+1} \quad \forall\, i \in \Omega, h \in P^-, \forall\, k \in K \tag{3.2e}$$

$$D_i + p_j \leqslant D_j \quad \forall\, (i,j) \in \Phi \tag{3.2f}$$

$$D_i + p_j - D_j \leqslant M(1 - Z_{ij}) \quad \forall\, (i,j) \in \Psi \setminus \Phi \tag{3.2g}$$

$$D_j + p_i - D_i \leqslant M Z_{ij} \quad \forall\, (i,j) \in \Psi \setminus \Phi \tag{3.2h}$$

$$D_i + t_{ij} + p_j - D_j \leqslant M(2 - X^k_{ih} - X^k_{j\,h+1}) \quad \forall\, i,j \in \Omega, \forall\, h \in P^-, \forall\, k \in K \tag{3.2i}$$

$$D_i - p_i \geqslant (r_k + t^k_{0i}) X^k_{i1} \quad \forall\, i \in \Omega, \forall\, k \in K \tag{3.2j}$$

$$D_i + t^k_{iT} - C^k \leqslant M\Big(1 - \sum_{h \in P} X^k_{ih}\Big) \quad \forall\, i \in \Omega, \forall\, k \in K \tag{3.2k}$$

$$X^k_{ih} \in \{0,1\} \quad \forall\, i \in \Omega, \forall\, h \in P, \forall\, k \in K \tag{3.2l}$$

$$Z_{ij} \in \{0,1\} \quad \forall\, (i,j) \in \Psi \setminus \Phi \tag{3.2m}$$

$$D_i, C^k \geqslant 0 \quad \forall\, i \in \Omega, \forall\, k \in Q \tag{3.2n}$$

In model (3.2), the objective function (3.2a) is identical with the objective function of the Kim and Park formulation. The vessel overall completion time is computed by constraints (3.2b). Constraints (3.2c) guarantee that each task is performed by such a QC. Constraints (3.2d) ensure that each position related to a QC

is assigned at most to one task. Constraints (3.2e) guarantee that the schedule of task assigned at each QC is constructed by the first empty position. The precedence relationship among task couples are defined by constraints (3.2f), while constraints (3.2g) and (3.2h) ensure non-simultaneity relations. The constraints (3.2i) guarantee that the tasks assigned to the same QC are performed without overlapping. Constraints (3.2j) compute the completion time of the first task performed by a QC taking into account *i*) the QC release time and *ii*) the time required to move from the QC initial position to the bay of the task. Constraints (3.2k) compute the completion time of the QCs considering the completion time of the last performed task and the time required to move from the bay position of the last task to the final position of the QC. At the end, constraints (3.2l)–(3.2n) are the constraints on the decision variables.

We suggest to set t^k_{0j} and t^k_{iT} to 0, $\forall\, i, j \in \Omega$ and $\forall\, k \in K$, assuming that each QC is located in front of the first task (i.e., after the release time the QC is exactly located in front of the first assigned task) and that the final position of the QC can be omitted.

Moreover, as we have shown in a research study on assignment and deployment of QC along the berth [41, 39], sometime is necessary to consider that a crane may have a *release time* and also a *due date*. In this context, the due date is the time in which a quay crane becomes un-available (e.g., because it has been assigned to another vessel). Therefore, let d_k be the due date of the crane $k \in K$, then we can introduce in model (3.2) the following constraints:

$$C^k \leqslant d_k \qquad \forall\, k \in K \tag{3.3}$$

Complexity examination Note that this model has $O(|\Omega|\cdot|P|\cdot|K|)$ variables and constraints. As we stated before on page 106, in the worst case $|P| = n$, therefore, in this case, models (3.1) and (3.2) have an identical complexity. Notwithstanding, using such a heuristic for computing a valid good enough or sub-optimal schedule, we can compute an upper-bound for the cardinality of P, and thus we can surely affirm that formulation (3.2) may have a lower complexity then formulation (3.1).

On this topic, we must consider that the Kim and Park formulation has been

widely studied in the past considering the analogy of the structure of model (3.1) with the well-known TSP problem [54]. Henceforth, some branch-and-bound and branch-and-cut algorithms have been proposed to solve it exactly.

Preliminary test on same instances using both formulations have been performed using the ILOG's CPLEX Solver 10.0 (tests were run on a 2.0 GHz Intel Pentium M computer with 2GB of memory).

The empirical results seem to prove that the formulation (3.2) is capable to find the optimal solution in less time than the formulation (3.1). Therefore, we use the formulation (3.2) to obtain problem solution of the set of instances defined in the section dedicated to numerical results (Section 3.5).

At this point, one may easily recognize that the MIP formulation focuses on the sole allocation/scheduling decisions to be taken within the more complex, dynamic discharge/loading process illustrated in Figure 3.1. A practical solution to the optimization of the overall logistic process is proposed, in the following, by resorting to simulation-based optimization.

3.4 Simulation-based Optimization Approach

In Chapter 2 we provided an introduction to *simulation-based optimization* (SO). Here, we recall that SO is well known as the optimization of expected performance measure(s) based on outputs from stochastic simulations of any given system/process, whose dynamic behavior is partially defined by some decision variables and constraints.

Here, the expected performance measure is the expected value of the overall completion time (i.e. the makespan) for the discharging/loading operations of a specific vessel. The makespan should be estimated through simulation of the queuing network model in Figure 3.1.

The formulation of the simulation-optimization problem would require to replace the objective function (3.2a) of problem (3.2) with the following $f(\theta) = E[L(\theta)]$, which also accounts for implicit additional process features and queuing phenomena

when searching for the optimal vector of decision variables θ. In the SO methods proposed in the following, solution "comparison" is based on statistics for the makespan which are computed on a certain number of observations. Since these observations are random variates returned from a simulation process, there are no guarantees of selecting the best design during the solution comparison, despite it being truly representative of the best system configuration. To this end, as stated in Chapter 2, at the "comparison step" of each algorithm we decided to use the Rinott's procedure [72] to perform a correct selection of a solution with at least a probability $1 - \alpha$ (with α small enough).

Ere now, we claimed that the optimization of the QCSP may be pursued via the stochastic simulation of the queuing network model depicted in Figure 3.1. Here the numerical experiments are organized as follow. First of all, to compare the performance of the metaheuristics NP, RBEES, and ABEES described in Chapter 2 with the solution provided by the commercial solver CPLEX, some procedure able to evaluate the makespan of a feasible solution for the QCSP must be presented. In particular, in Section 3.4.1 we present a procedure for the makespan evaluation of the QCSP, for both deterministic and stochastic evaluation of a feasible schedule. The procedure evaluate a schedule without considering the effects of blocking and starvation phenomenons due to the presence of the buffer area under each crane and straddle carriers in the real system. Successively, we show the effects of stochastic phenomena on the evaluation of the performance of a solution by means of an optimization via simulation approach.

Then, in Section 3.4.2 a distance measure for the QCSP is proposed. The distance measure is useful in the development of the ABEES search scheme.

Finally, before presenting numerical results, we present in Section 3.4.3 a description of the procedures adopted for the generation of new global and local solutions. This procedure are adopted in both RBEES and ABEES, while the first procedure is adopted only within the NP scheme.

3.4.1 Evaluation of the makespan

We have developed a simple algorithm to evaluate the makespan of a solution $\mathcal{S}$ for the QCSP, which can be also generalized in order to simulate a schedule under uncertainty (i.e. generating the tasks processing time sampling by such distribution).

In the following, we recall to the notations defined by page 103. The solution $\mathcal{S}$ is a set of m schedules of tasks. The schedule of tasks that refers to the the QC $i \in K$ is called $\mathcal{S}_i$.

The algorithm, called *quasi-simulated makespan evaluation*, is the following:

Initialization Let MS be the makespan of the assigned schedule.

For each task $i \in \Omega$ let be: bl_i the location of the bay whereat task i is referred; and, we recall that p_i is the processing time of the task i.

For each QC $j \in K$, let be: $\mathcal{S}_j$ the schedule of assigned tasks; n_j the index of the next task that must be processed by the QC j, with n_j equal initially to the first task in $\mathcal{S}_j$ and $\mathcal{S}_j = \mathcal{S}_j \setminus \{n_j\}$; s_j and cl_j be respectively the travel speed and the current location of j, afterwards set cl_j equal to the location of the first task that must be performed, i.e. $cl_j = bl_{n_j}$; t_i be the clock of the i, with $t_i = 0$.

Moreover, let $\mathcal{T}$ be the list of QCs' clock, K^W be the list of QCs that are currently working (i.e. are performing a task and moving at task location), K^S be the list of QCs that are currently starved (i.e. are waiting for the availability of the next assigned task), $\mathcal{T}^W$ be the list of currently working QCs' clock. Finally, let K^C be the set of QCs that have completed the assigned schedule $\mathcal{S}_j$ (i.e. the set of QCs that have processed all the assigned tasks), with $K^C = \emptyset$.

Step 1 Set $\mathcal{T} = \{t_j | j \in K \setminus K^C\}$, $K^W, K^S, \mathcal{T}^W, \mathcal{T}^S = \emptyset$.

Goto *Step 2*.

Step 2 If $\mathcal{T} \setminus \mathcal{T}^W = \emptyset$, then goto *Step 1*.

Else compute $\hat{\jmath} = \underset{j \in K}{arg\,min} \{\mathcal{T} \setminus \mathcal{T}^W\}$.

Goto *Step 3*.

Step 3 If $n_{\hat{\jmath}}$ can be processed without constraints violation (i.e., *precedence* and *non-simultaneity* constraints), then:

1. $K^W = K^W \cup \{\hat{\jmath}\}$, and $\mathcal{T}^W = \mathcal{T}^W \cup \{t_{\hat{\jmath}}\}$.
2. Set $t_{\hat{\jmath}} = t_{\hat{\jmath}} + s_{\hat{\jmath}} \cdot |cl_{\hat{\jmath}} - bl_{n_{\hat{\jmath}}}| + p_{n_{\hat{\jmath}}}$.
3. If $\hat{\jmath} = \underset{j \in K^W}{arg\ min}\{\mathcal{T}^W\}$ then,
   for each $i \in K^S$, set $t_i = t_{\hat{\jmath}}$.
4. If $\mathcal{S}_{\hat{\jmath}} \neq \emptyset$, then set $n_{\hat{m}}$ equal to the index of the first task in $\mathcal{S}_{\hat{\jmath}}$ and set $\mathcal{S}_{\hat{\jmath}} = \mathcal{S}_{\hat{\jmath}} \backslash \{n_{\hat{\jmath}}\}$;
   else $K^C = K^C \cup \{\hat{\jmath}\}$, and goto *Step 4*.

Else, $K^S = K^S \cup \{\hat{\jmath}\}$, and $\mathcal{T}^S = \mathcal{T}^S \cup \{t_{\hat{\jmath}}\}$. Finally, goto *Step 2*.

Step 4 If $K^C = K$, then set $MS = \underset{j \in K}{arg\ max}\{t_j\}$ and *STOP*: Present MS as the makespan.

Else, goto *Step 1*.

The preceding procedure computes the makespan of the QCSP with precedence and non-simultaneity constraints. Nevertheless, the preceding procedure is valid for each $P|prec|C_{max}$ scheduling problem with precedence constraints among jobs and it can be modified in order to compute the makespan of $K|prec|C_{max}$ and $R|prec|C_{max}$ scheduling problems.

3.4.2 A distance measure

Many recent metaheuristics (e.g. the *Scatter Search*) use distance measures primarily for diversification purposes, i.e. to guide the search into previously unexplored regions of the search space. As described in Chapter 2, the ABEES metaheuristics needs a distance measure between solutions of an optimization problem. The development of effective distance measures however is not a trivial task [85]. In fact, a good distance measure should provide an accurate estimate of the difference (or similarity) between two given solutions.

Different solution representations require different distance measures. Scheduling problems have solutions that are naturally represented as a permutation of a set of

tasks and are also characterized by the fact that the order in which the tasks appear in the solution is very important. For this class of problems there is no agreed upon a common distance measure.

Ronald [74] proposes the *exact match distance*, that is an extension of the *Hamming distance*, and considers the exact position of a task in the solution to be important. Given two schedules $\mathcal{R} = [r_1, r_2, \ldots, r_n]$ and $\mathcal{S} = [s_1, s_2, \ldots, s_n]$ defined over a set of n tasks, the exact match distance $d_{EM}(\mathcal{R}, \mathcal{S})$ is defined as follows.

$$d_{EM}(\mathcal{R}, \mathcal{S}) = \sum_{i=1}^{n} m_i(\mathcal{R}, \mathcal{S}) \tag{3.4}$$

$$\text{where} \qquad m_i(\mathcal{R}, \mathcal{S}) = \begin{cases} 0 & \text{if } r_i = s_i, \\ 1 & \text{otherwise.} \end{cases}$$

This distance measure can be used to compare two different schedules of a single machine scheduling problem. In multiple machine scheduling, each machine has an own schedule defined over the set of tasks, therefore two solutions may differ by *i*) a permutation of the sequence of the tasks assigned at each machine and *ii*) the number of tasks assigned to each machine schedule. In particular, in the QCSP we are dealing with a *m-parallel machines scheduling problem*, therefore we are interested in constructing a distance measure that is able to compare the solutions $\mathcal{R}$ and $\mathcal{S}$, where $\mathcal{R} = \{\mathcal{R}_k | \forall\, k \in K\}$ and $\mathcal{S} = \{\mathcal{S}_k | \forall\, k \in K\}$, and $\mathcal{R}_k$ is the schedule of tasks assigned to the QC $k \in K$ in the solution $\mathcal{R}$ (and likewise for $\mathcal{S}_k$).

First and foremost, is necessary to consider that is possible to exchange the schedules assigned to the RTGs $i, j \in K$ into the same solution without affecting the makespan. Thus, in a *two*-cranes QCSP, we can also consider to be equivalent two solutions $\mathcal{R}$ and $\mathcal{S}$ where the schedules $\mathcal{R}_1$ and $\mathcal{R}_2$ are identical respectively to the schedules $\mathcal{S}_2$ and $\mathcal{S}_1$. In that way, extending the previous statement, two solutions $\mathcal{R}$ and $\mathcal{S}$ of a m-cranes QCSP are equivalent if each schedule in $\mathcal{R}$ is found in $\mathcal{S}$ assigned to whatsoever QC (i.e. in whatever order).

Then, verified that the solutions $\mathcal{R}$ and $\mathcal{S}$ are not equivalent, we can compute the distance between them using a method based on the exact match distance. This distance method for the QCSP problem with RTGs is based on the idea that the

distance must be computed between the most similar couple of schedules $(\mathcal{R}_i, \mathcal{S}_j)$ with $\mathcal{R}_i \in \mathcal{R}$ and $\mathcal{S}_j \in \mathcal{S}$, and not between the schedules assigned to the same QC in the solutions $\mathcal{R}$ and $\mathcal{S}$. This is reasonable if we recall to the possibility of exchanging the schedules assigned to the QCs into the same solution without changing the makespan: in fact, we can reorder the schedules in a solution without changing the makespan of the solution. Then, we are interested to identity the permutation of $\mathcal{R}$ that produces the minimum distance with the solution $\mathcal{S}$.

The following algorithm, called the *minimum exact match distance*, is used to compute the distance between two solutions of the QCSP.

Initialization Let d be the distance between two solutions $\mathcal{R}$ and $\mathcal{S}$ of a n-tasks m-RTGs QCSP. Let $\mathcal{R}_i$ and $\mathcal{S}_i$ be respectively the schedule of tasks assigned to the i-th RTG.

Step 1 For each schedule $\mathcal{R}_i \in \mathcal{R}$, if $\mathcal{R}_i \notin \mathcal{S}$, then goto *Step 2*.
Present $d = 0$ as the estimate of the distance between solutions $\mathcal{R}$ and $\mathcal{S}$.

Step 2 Let $\mathcal{S}^i$ be the i-th permutation of S, achieved by permuting the m schedules in $\mathcal{S}$ (thus, we have $m!$ permutations of $\mathcal{S}$). Finally, let d_i be the distance between the solution $\mathcal{R}$ and the i-th permutation of $\mathcal{S}$.
Then, for each $\mathcal{S}^i$ compute d_i as follows:

$$d_i = \sum_{j=1}^{m} d_{EM}(\mathcal{R}_j, \mathcal{S}_j)$$

where $d_{EM}(\mathcal{R}_j, \mathcal{S}_j)$ is the exact match distance (see equation (3.4) on page 113) between the schedules in position j into $\mathcal{R}$ and $\mathcal{S}$.
If two schedules $\mathcal{R}_j$ and $\mathcal{S}_j$ have different cardinality, then the distance $d_{EM}(\mathcal{R}_j, \mathcal{S}_j)$ is computed on the first $c_{\min}$ tasks, where $c_{\min} = \min\{|\mathcal{R}_j|, |\mathcal{S}_j|\}$. Then the distance $d_{EM}(\mathcal{R}_j, \mathcal{S}_j)$ is increased by $c_{\max}$, where $c_{\max} = \max\{|\mathcal{R}_j|, |\mathcal{S}_j|\}$.
Goto *Step 3*.

Step 3 *STOP*: Present $d = \min_{i=1,\ldots,m!}\{d_i\}$ as the distance between $\mathcal{R}$ and $\mathcal{S}$.

The following example shows how is computed the distance between two solutions of a QCSP using the minimum exact match distance. Consider two solutions $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2\}$ and $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2\}$ for a *five*-tasks and *two*-RTGs QCSP. Let be $\mathcal{R}_1 = [1, 2, 3]$, $\mathcal{R}_2 = [4, 5]$, $\mathcal{S}_1 = [5, 4]$ and $\mathcal{S}_2 = [1, 2, 3]$. The distance computed by comparing the schedules $\mathcal{R}_1$ with $\mathcal{S}_1$ and $\mathcal{R}_2$ with $\mathcal{S}_2$ is respectively equal to 3 for the first couple and 2 for the second couple, i.e. the total distance is 5. Using the preceding algorithm, the minimum distance is obtained comparing the most similar couple of schedules, i.e. $\mathcal{R}_1$ with $\mathcal{S}_2$ (distance 0) and $\mathcal{R}_2$ with $\mathcal{S}_1$ (distance 2), therefore the total distance is 2.

3.4.3 Generation of a feasible schedule

In this Section we aim to describe the procedures used to perform global and local search within the BEES framework and for the partitioning of the feasible region. Here, the feasible region Θ is a finite set of n-dimensional points. The point $\theta \in \Theta$ (i.e. a *feasible solution*) is a valid schedule for the QCSP, where $\theta = (\theta_1, \ldots, \theta_n)$ and $\theta_i \in \Theta_i$ depicts a valid assignment of the task $i \in \Omega$, and Θ_i is the finite set of possible values of θ_i (e.g., the task i is scheduled in position p on the crane k). The cardinality of Θ_i changes in function of the assignment of the other tasks, i.e., it changes solution by solution. This particular property of Θ_i will be explained later and is due to the precedence relationship among tasks. As introduced in Section 2.4, the number of values that θ_i can assume is $|\Theta_i|$.

Here we recall that a solution is the sets of task-sequence assigned to each QC. Therefore, as proposed in Sections 3.4.1 and 3.4.2, a solution/schedule $\mathcal{S}$ is a set of valid assignment task-crane, where for each task assigned to a crane, the order in which the crane must perform the assigned tasks is specified. Let $\mathcal{S}_k$ be the ordered list of tasks assigned to QC $k \in K$ and $s_{k,p}$ be the task scheduled in position p on crane k. Therefore, a schedule is depicted as follows:

$$\mathcal{S} = \{\mathcal{S}_k = [s_{k,1}, \ldots, s_{k,n_k}] \,|\, \forall\, k \in K, \sum\nolimits_{k \in K} n_k = |\Omega|\}.$$

The procedures proposed in the following Sections have the goal to describe the procedures used for the development of the metaheuristics adopted in this Chapter.

Generally, the set of precedence constraints Φ includes some implicit precedence constraints. In fact, according to the famous *transitive property*, by two precedence constraints $(i,k),(k,j) \in \Phi$ is possible to infer the precedence constraint (i,j). The mathematical models (3.1) and (3.2) take into account these implicit precedence constraints through their structure. Otherwise, the dynamic random-construction of a feasible schedule that only considers the order in which each task will be processed by the a QC, need the explicit generation of these "new" constraints. Therefore, before starting the search process, a pre-processing on the set Φ is required, as proposed in the following procedure.

Initialization Let η be the current iteration, then set $\eta = 0$.
Let c_i and c_n be respectively the *initial* and the *new* cardinality of Φ. Let Φ^η be the set of precedence constraints inferred at iteration η.

Step 1 Set $\eta = \eta + 1$, $\Phi^\eta = \emptyset$ and $c_i, c_n = |\Phi|$.
For each $(i,k),(k,j) \in \Phi$, then set $\Phi^k = \Phi^k \cup \{(i,j)\}$.
Goto *Step 2*.

Step 2 . Set $\Phi = \Phi \cup \Phi^\eta$ and $c_n = |\Phi|$.
If c_n is not equal to c_i, then goto *Step 1*. Else *STOP*: Present Φ as the new set of precedence constraints.

The procedures for global and local search adopted in the RBEES and ABEES search schemes are proposed in Sections 3.4.4 and 3.4.5. The partitions procedure adopted in the NP algorithms is proposed in Section 3.4.6.

3.4.4 Global search procedure

The global search procedure described in the following aims to sample a feasible solution from the solutions set Θ. The procedure iteratively randomly select a not yet assigned task $i \in \Omega$ and try to assign it at the position p of the schedule of

a crane k, where p and k are chosen randomly. The assignment must produce a feasible solution θ.

To generate a feasible schedule, the procedure must check that the assignment of i at position p in the schedule of crane k satisfies all the precedence constraints in Φ. Nevertheless, the satisfaction of the precedence constraints not necessarily produce a feasible schedule. In fact, a solution produced checking only the constraints in Φ may lead the process of evaluation of a solution θ to *deadlock phenomena*. The following example shows how a solution with only two precedence constraints could be infeasible (i.e., it produces a deadlock), notwithstanding all the precedence constraints are satisfied.

Consider a solution $\mathcal{S}$ for a n-tasks and *two*-RTGs QCSP. Suppose that i) the set of precedence constraints is $\Phi = \{(i,j),(o,p)\}$, and ii) the schedule of tasks assigned to the first crane is $\mathcal{S}_1$, with $|\mathcal{S}_1| = n_1$, and the schedule of tasks assigned to the second crane is $\mathcal{S}_2$, with $|\mathcal{S}_2| = n_2$ and $n_1 + n_2 = n$, where:

$$\mathcal{S}_1 = [s_{1,1}, s_{1,2}, \ldots, s_{1,\alpha}, \ldots, s_{1,\beta}, \ldots, s_{1,n_1}],$$
$$\mathcal{S}_2 = [s_{2,1}, s_{2,1}, \ldots, s_{2,\gamma}, \ldots, s_{2,\epsilon}, \ldots, s_{2,n_2}].$$

Imagine that the tasks scheduled in position $s_{1,\alpha}$ and $s_{1,\beta}$ are respectively the task j and the task o, while the tasks scheduled in position $s_{2,\gamma}$ and $s_{2,\epsilon}$ are respectively the task p and the task i. Therefore, task j must be performed before task o by the first crane, and task p must be performed before task i by the second crane.

It is easy to recognize that the solution composed by the schedules $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfies both precedence constraints. But, the makespan of the proposed solution can not be evaluated: In fact, according to the precedence constraint (i,j), the task j can not be processed by the first crane once task i is performed by the second crane; In a similar way, to perform the task i, the second crane must process task p, that can not be performed because, according to precedence (o,p), the task o will not be processed by the first crane until the predecessor task j will be completed. Definitely, this is a typical deadlock situation. To avoid this situations, during the construction of a solution, is necessary to take into account all the precedence constraints in Φ, and, moreover, a set of precedence constraints that are built at

runtime considering the current structure of the proposed solution. Let $\Phi(\mathcal{S})$ be the set of runtime constraints derived by the solution $\mathcal{S}$ joined the known "static" precedence constraints in Φ.

The following procedure is designed to generate the set $\Phi(\mathcal{S})$ for a specific (partial) schedule.

Initialization Let η be the current iteration, then set $\eta = 0$.

Let Φ be the set of precedence constraints (possibly pre-processed using the procedure at page 116) and $\Phi(\mathcal{S})$ be the set of runtime precedence constraints, then set $\Phi(\mathcal{S}) = \Phi$. Let Φ^η be the set of precedence constraints inferred at iteration η. Let c_i and c_n be respectively the *initial* and the *new* cardinality of $\Phi(\mathcal{S})$.

Let $\mathcal{S} = \{\mathcal{S}_k = [s_{k,1}, \ldots, s_{k,n_k}] \,|\, \forall\, k \in K, \sum_{k \in K} n_k \leqslant |\Omega|\}$ be a partial assignment of a sub-set of the n tasks in Ω to the q cranes in K.

Step 1 Set $\eta = \eta + 1$, $\Phi^\eta = \emptyset$ and $c_i, c_n = |\Phi(\mathcal{S})|$.

For each $\mathcal{S}_k \in \mathcal{S}$ and for all $s_{k,\alpha}, s_{k,\beta} \in \mathcal{S}_k$ with $s_{k,\alpha} < s_{k,\beta}$, then:

- Let i be the task $s_{k,\alpha}$ and j be the task $s_{k,\beta}$.
- For all $(o,i),(j,p) \in \Phi(\mathcal{S})$,
  if $(o,p) \notin \Phi(\mathcal{S}) \cup \Phi^\eta$ with $o \neq p$, then set $\Phi^\eta = \Phi^\eta \cup \{(o,p)\}$.

Goto *Step 2*.

Step 2 . Set $\Phi(\mathcal{S}) = \Phi(\mathcal{S}) \cup \Phi^\eta$ and $c_n = |\Phi(\mathcal{S})|$.

If c_n is not equal to c_i, then goto *Step 1*. Else *STOP*: Present $\Phi(\mathcal{S})$ as the set of precedence constraints for the (partial) solution $\mathcal{S}$.

The global search procedure here proposed aims to produce a feasible solution by starting from a schedule $\mathcal{S}$ and iteratively assigning all the un-assigned tasks in the available positions of the q lists $\mathcal{S}_k, k \in K$ by considering the set of precedence $\Phi(\mathcal{S})$. At each iteration, the procedure randomly selects a un-assigned task and try to assign it in a position of a randomly selected crane.

The schedule $\mathcal{S}$ may also include a valid assignment of a sub-set of Ω (this feature is useful for the generation of a solution in the NP metaheuristics).

The global search procedure is the following.

Initialization Let Ω^u be the set of un-scheduled tasks and $\mathcal{S}$ be the global sampled solution, then set $\Omega^u = \Omega \backslash \{s_{k,p} \in \mathcal{S}_k | k \in K, 1 \leqslant p \leqslant n_k\}$. Compute the set $\Phi(\mathcal{S})$ by the set of precedence constraints Φ, the current solution $\mathcal{S}$, and using the procedure proposed above. Finally, let VB and VA be two boolean flags.

Step 1 Let i be a task randomly extracted by Ω^u, k be a crane randomly extracted by K.
Goto *Step 2*.

Step 2 Compute $n_k = |\mathcal{S}_k|$.
If n_k is equal to 0, then:

- Add i in first position in $\mathcal{S}_k$.
- Set $\Omega^u = \Omega^u \backslash \{i\}$.
- Goto *Step 1*.

Else:

- Let p be a position randomly selected within the range $[1, n_k]$.
- Update the set $\Phi(\mathcal{S})$.
- Goto *Step 3*.

Step 3 Set $VB = false$. For all $s_{k,\alpha}, \in \mathcal{S}_k$ with $s_{k,\alpha} < s_{k,p}$, then:

- Let j be the task $s_{k,\alpha}$.
- If exists $(i, j) \in \Phi(\mathcal{S})$, then set $VB = true$ and goto *Step 4*.

Goto *Step 4*.

Step 4 Set $VA = false$. For all $s_{k,\alpha}, \in \mathcal{S}_k$ with $s_{k,\alpha} \geqslant s_{k,p}$, then:

- Let j be the task $s_{k,\alpha}$.
- If exists $(j, i) \in \Phi(\mathcal{S})$, then set $VA = true$ and goto *Step 5*.

Goto *Step 5*.

Step 5 [*Some precedence constraints has been violated*]
If both VB and VA are $true$, then goto *Step 1*.
Else if VB is $true$, then:

- If p is equal to n_k, then goto *Step 1.*
- Else set $p = p + 1$;

Else if VA is $true$, then:

- If p is equal to 1, then goto *Step 1.*
- Else set $p = p - 1$;

Goto *Step 3.*

Step 6 [*No precedence constraints has been violated*]
Insert i in position p in the list $\mathcal{S}_k$.
Set set $\Omega^u = \Omega^u \backslash \{i\}$.
Goto *Step 7.*

Step 7 If $\Omega^u = \emptyset$, then STOP: Present $\mathcal{S}$ as a valid global sampled solution. Else, goto *Step 1.*

The previous procedure checks if a task i can be inserted in position p of a list $\mathcal{S}_k$ by evaluating if some precedence constraint is violated before or after the specified position. In particular, a precedence constraint is "violated before" if exists a precedence constraint $(i, j) \in \Phi(\mathcal{S})$ and the task j is assigned to the crane k in a position preceding p. On the contrary, a precedence constraint is "violated after" if exists a precedence constraint $(j, i) \in \Phi(\mathcal{S})$ such as the task j is assigned to the crane k in a position that is greater or equal to p. Note that, the set of precedence constraints $\Phi(\mathcal{S})$ must be computed at each iteration and it depends on the current structure of the solution $\mathcal{S}$. The boolean flags VB, VA are used to check if some constraints is violated before/after the randomly selected position p.

3.4.5 Local search procedure

The local search procedure proposed in this Section is used in the RBEES and ABEES search schemes for random sampling feasible solutions within the neighborhood of a promising solution θ. According to the concept of distance between two schedules $\mathcal{R}$ and $\mathcal{S}$ that we suggested in Section 3.4.2, the procedure that we are proposing sample new points whose distance measure is at most 2. Thus, a neighbor of a solution $\mathcal{S}$ can be generated by selecting two crane-list $\mathcal{S}_i$ and $\mathcal{S}_j$ over the

set of cranes, and: *i*) if $i = j$, performing an exchange move between the task at position h and that at position k in the same list $\mathcal{S}_i$ (this is the only move that can be performed on a same crane-schedule); *ii*) if the list $\mathcal{S}_i$ is empty, removing the task at the tail of $\mathcal{S}_j$ and adding it at the tail of $\mathcal{S}_i$ (and *vice versa* if $\mathcal{S}_j$ is empty); *iii*) if the list $\mathcal{S}_i$ and $\mathcal{S}_j$ are not empty, performing an exchange move between the task at position h in $\mathcal{S}_i$ and the task in position k in $\mathcal{S}_j$ (not that if $h = k$ the move produce two different solution at the minimum distance). Obviously, almost one of the two list $\mathcal{S}_i$ and $\mathcal{S}_j$ must have a non-null cardinality and a move is executed if and only if no precedence constraint is violated (considering the set of precedence constraints Φ opportunely pre-processed).

The next procedure is used to perform the neighbors generation.

Initialization Let η be the current iteration and η^{max} be the maximum number of iterations, then set $\eta = 0$ and η^{max} big enough.
Let $\mathcal{S}$ be a feasible solution, and $\mathcal{R}$ be a neighbor of $\mathcal{S}$, then set $\mathcal{R} = \mathcal{S}$. Thus, let $\Phi(\mathcal{R})$ be the set of precedence constraints built on the solution $\mathcal{R}$.

Step 1 Set $\eta = \eta + 1$. Let k_1 and k_2 be cranes randomly extracted by K, with even $k_1 = k_2$.
Goto *Step 2*.

Step 2 [*Task swapping on the same crane schedule*]
If $k_1 = k_2$ then, for each couple of positions until all the positions have been examined or a solution has been found.

- Let $r_{k_1,p}$ and $r_{k_1,q}$ be two positions with $p \neq q$.
- Let i be the task at position p and j that in position q.
- Set i in position $r_{k_1,q}$ and j in position $r_{k_1,p}$. Compute $\Phi(\mathcal{R})$ as described in Section 3.4.4.
  For each $\mathcal{R}_k \in \mathcal{R}$ and for all $r_{k,\alpha}, r_{k,\beta} \in \mathcal{R}_k$ with $r_{k,\alpha} < r_{k,\beta}$, then:
  - Create a boolean flag $FEASIBLE$, and set its value to *true*.
  - Let v be the task $r_{k,\alpha}$ and w be the task $r_{k,\beta}$.

– If exists $(w, v) \in \Phi(\mathcal{R})$, then re-set i in position $r_{k_1,p}$ and j in position $r_{k_1,q}$ and set $FEASIBLE = false$.

If $FEASIBLE = true$ goto *Step 4*.

Else goto *Step 3*.

Step 3 [*Task swapping on a couple of tasks from two schedule*]

- For each couple of positions $r_{k_1,p}$ and $r_{k_2,q}$ until all the couple have been examined or a solution has been found.
- Let i be the task at position p on crane k_1 and j that in position q on crane k_2.
- Set i in position $r_{k_2,q}$ and j in position $r_{k_1,p}$. Compute $\Phi(\mathcal{R})$ as described in Section 3.4.4.
  For each $\mathcal{R}_k \in \mathcal{R}$ and for all $r_{k,\alpha}, r_{k,\beta} \in \mathcal{R}_k$ with $r_{k,\alpha} < r_{k,\beta}$, then:
  – Create a boolean flag $FEASIBLE$, and set its value to $true$.
  – Let v be the task $r_{k,\alpha}$ and w be the task $r_{k,\beta}$.
  – If exists $(w, v) \in \Phi(\mathcal{R})$, then re-set i in position $r_{k_1,p}$ and j in position $r_{k_2,q}$ and set $FEASIBLE = false$.

  If $FEASIBLE = true$ goto *Step 4*.

[If a list is eventually empty, then remove the tail of the non-empty list a try to add it to the empty list] Goto *Step 4*.

Step 5 If distance between $\mathcal{R}$ and $\mathcal{S}$ is greater than 0, then *STOP*: Present $\mathcal{R}$ as a neighbor of $\mathcal{S}$.
Else if $\eta < \eta^{max}$ goto *Step 1*.
Else if η is equal to η^{max}, then *STOP*: No neighbor solution has been found.

In some rare cases, e.g., for strongly constrained problems (i.e. with a high number of precedence constraints with respect to the number of tasks), is possible that no solutions with a distance of 2 may be computed with respect to a schedule $\mathcal{S}$.

Therefore, the previous procedure tries to generate a neighbor of $\mathcal{S}$ until a maximum number of attempts (η^{max}) is reached (this is an user-defined parameter).

3.4.6 Partitioning scheme

Here we describe how the Nested Partitions metaheuristics has been tailored on the QCSP, considering the description of the framework provided in Section 2.4.

We recall that a point $\theta \in \Theta$ has n dimensions (one for each task), i.e., $\theta = (\theta_1, \ldots, \theta_n)$ and $\theta_i \in \Theta_i$ depicts a valid assignment of the task $i \in \Omega$. We also stated that Θ_i is the finite set of possible values of θ_i and that, the cardinality of this set of values is a number $M_1(\sigma)$ which take into account the decision variable θ_i and the region σ currently partitioned by the NP method. The NP partition scheme is based on a known a priori partition scheme, or rather, it use an ordered list of the θ_i decision variables. The order of the θ_i values is produced in some way, also randomly. Thus, the partitioning scheme is used to know on which decision variable θ_i the algorithm must partition the current most promising region. The current most promising region σ can be partitioned in exactly $M_i(\sigma)$ subregions. Once a partition have been performed, a certain number of points must be sampled for each subregion. Each subregion corresponds to a "partial" solution, i.e., for all the already partitioned variables, some feasible assignment of the related tasks has been performed. Therefore, points may be sampled assigning the remaining un-partitioned tasks.

Thus, in the following we provides two procedure: the first is used to determine the value $M_i(\sigma)$ for each θ_i in a desired region σ, while the second is used to sample points from a subregion. The notation used in the following have been introduced in Section 2.4.

Initialization Let $\mathcal{S} = \{\mathcal{S}_k = [s_{k,1}, \ldots, s_{k,n_k}] \,|\, \forall\, k \in K, \sum_{k \in K} n_k \leqslant |\Omega|\}$ be a partial assignment of a sub-set of the n tasks in Ω to the q cranes in K. Therefore, let $\sigma(\eta)$ be the current most promising region at the η-th iteration of the NP-method, then $\mathcal{S} \equiv \sigma(\eta)$.
Let $M_i(\mathcal{S})$ be the number of values that the decision variable θ_i can assume in a region σ, then set $M_i(\mathcal{S}) = 0$.

Moreover, let $\Phi(\mathcal{S})$ be the set of precedence constraints built on the solution $\mathcal{S}$.

Step 1 If $\mathcal{S}$ is "empty" (i.e., $\mathcal{S}_k = \emptyset \; \forall \, k \in K, \forall \, j \in \mathcal{S}$, then:

- Set $M_i(\mathcal{S}) = q$, where $q = |K|$.
- Goto *Step 3*.

Else goto *Step 2*.

Step 2 Compute $\Phi(\mathcal{S})$ as described in Section 3.4.4.
For each $\mathcal{S}_k$ and for each possible position $p = 1, \ldots, n_k + 1$, then:

- Let V be a boolean flag, then set $V = false$.
- For all $s_{k,\alpha}, \in \mathcal{S}_k$ with $s_{k,\alpha} < s_{k,p}$, then:
  - Let h be the task $s_{k,\alpha}$.
  - If exists $(i, h) \in \Phi(\mathcal{S})$, then set $V = true$.
- If V is $false$, then set $M_i(\mathcal{S}) = M_i(\mathcal{S}) + 1$.

Goto *Step 3*.

Step 3 *STOP*: Present $M_i(\mathcal{S})$ as the number of possible values of θ_i in region identified depicted by $\mathcal{S}$.

At this point, once the cardinality of Θ_i has been computed, at the η-th iteration the NP method samples a set of points from each of the subregion $\sigma_j(\eta)$, with $j = 1, \ldots, M_i(\sigma(\eta))$. Therefore, two key aspect are the procedures used to sample a point by a subregion and by the surrounding region.

We recall that a each subregion can be see as a partial assignment of the tasks in Ω, i.e. it is a schedule $\mathcal{S} = \{\mathcal{S}_k | \forall \, k \in K, \sum_{k \in K} |\mathcal{S}_k| < |\Omega|\}$. Therefore, using the global sampling procedure proposed in Section 3.4.4, a feasible point θ can be sampled by the subregion related to the partial schedule $\mathcal{S}$. In fact, the global sampling procedure that we proposed is able to complete a partial solution by producing a feasible scheduling for all the un-assigned tasks.

At the η-th iteration, the NP implementation that we are proposing samples a point by the surrounding region in two (iterative) phases. In the first phase, a new feasible point θ is sampled from Θ. Subsequently, the distance measure proposed in

Section 3.4.2 is adopted to compute the distance between θ and the current most promising region $\sigma(\eta)$. If the distance is not maximum, then there is some task in θ that has been scheduled in the same position on the same crane within the partial solution depicted by $\sigma(\eta)$.

3.5 Numerical experiments

The object of the analysis reported in the following is twofold. On one hand, experiments on the QCSP mean to investigate and compare the performance of the RBEES, ABEES and NP method, when system dynamics are affected by some major source of uncertainty: *i*) the quay crane travel speeds and *ii*) the discharge/loading service times operated by the quay cranes. The results returned are also examined in relation to the optimal value found by the ILOG's commercial software CPLEX for the optimization model proposed in Section 3.3.2, which provides a lower bound on the value of the *makespan* when data is deterministic. On the other hand, the same tests intend to show how SO algorithms are often the only practical solution method available when dealing with difficult-to-solve combinatorial problem instances, embedded in realistic, dynamic environments characterized by several elements of randomness.

The QCSP is an operative problem that arises in every maritime container terminal. It is generally solved by means of deterministic optimization. This approaches do not consider that in the short-term the effects of stochastic processes must not be ignored. Here, we show the results achieved on 13 problem instances using the CPLEX solver, and the metaheuristics RBEES, ABEES and Nested Partitions (NP) with deterministic discharging/loading and traveling time for the QCs. In Table 3.1 are reported the main features of the instances used in the numerical experiments (Table 3.2 groups together similar instances). A detailed description about the instances and their development is reported in Appendix B.

Instance	*Number of QCs*	*Number of tasks*	*Number of bays*	$\lvert\Phi\rvert$	$\lvert\Psi\backslash\Phi\rvert$
1	2	10	10	2	5
2	2	10	10	4	12

Instance	*Number of QCs*	*Number of tasks*	*Number of bays*	$\lvert\Phi\rvert$	$\lvert\Psi\backslash\Phi\rvert$
3	2	10	10	3	6
4	2	15	10	7	14
5	2	15	10	5	22
6	2	15	10	7	16
7	3	20	20	7	16
8	3	20	20	8	13
9	3	20	20	9	10
10	3	25	20	9	18
11	3	25	20	9	23
12	3	25	20	10	24
13	3	24	8	16	48

Table 3.1: Instances for the QCSP.

Instances	*Number of QCs*	*Number of tasks*	*Number of bays*	$\overline{\lvert\Phi\rvert}$	$\overline{\lvert\Psi\backslash\Phi\rvert}$
$\{1,\ldots,3\}$	2	10	10	3	8
$\{4,\ldots,6\}$	2	15	10	6	17
$\{7,\ldots,9\}$	3	20	20	8	13
$\{10,\ldots,12\}$	3	25	20	9	22
13	3	24	8	16	48

Table 3.2: Summarized description of the instances for the QCSP.

We must clarify that we assumed that the QCs were always available in all the instances and that the initial location of each QC was identical to the bay-location of the first task that must be performed.

For explanatory purposes, the *instance 13* described in Appendix B is an example of a real vessel and is depicted in Figure 3.2. The vessel is able to carry up to 1,200 TEUs. The reader can argue about the size of real problems: in fact, the largest containership is actually able to carry 11,000 TEUs, therefore the QCSP is a very complex problem to face.

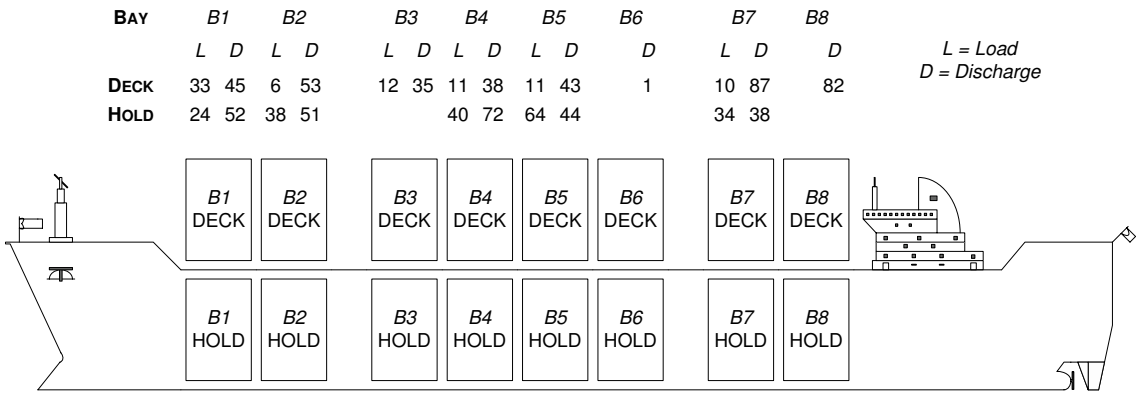


Figure 3.2: A real discharging/loading problem.

3.5.1 Choice of the parameters

The first step to deal with before starting the experiments is the set-up of the three metaheuristics. The evaluation of the parameters of each metaheuristics is generally a key aspect related to the core process of searching the best solution. Here, we need to evaluate the performance of the following parameters.

The RBEES search scheme uses the p parameter, that is the probability value depending on the global or local sampling procedure is used at each iteration of the algorithm. In particular, with probability p the global sampling procedure is used, otherwise the local sampling procedure is adopted.

The ABEES search scheme is more complex than the RBEES scheme. In particular, several parameters must be taken into account: k_l and k_g are respectively the number of points samples using the local and global search procedure at each stage of the search process; d and δ are two thresholds, namely the distance threshold d and the improvement threshold δ, with, for this problem, $d \in \mathbb{N}$.

Finally, in the BEES framework, a common parameter for both algorithms RBEES and ABEES is η^{max}, that is a parameter used within the stopping criterion. In particular, the search schema is stopped if no substantial improvement has been found within η^{max} iterations.

For what concern the NP method, three main parameters must be addressed. The first one regard the number of k^{max} of singleton point that must be selected

using the NP method (see the scheme at page 84). Then, the number n_1 of i sets $\mathcal{D}_j^i$ required in the first sampling stage for the region j, and the number N of point sampled from the region j for each set $\mathcal{D}_j^i$ must be defined. We recall that n_1 is, here, the "number of observations" from each examined region in the first stage of the Rinott's procedure. The other parameters for the Rinott's procedure adopted within the NP method are the α and δ, with for all the instances: $\alpha = 0.05$ and $\delta = 0.25$ (the value of the *Rinott's constant* h is computed using the tables in [92]).

Here, the partitioning schema (i.e., the order in which the decision variables are selected) is chosen at random.

We tested the RBEES algorithm and the ABEES with the following parameters $\eta^{max} \in \{1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000\}$. The overall number of iterations has been set to $10,000$.

For the RBEES, we test the metaheuristics with $p \in \{0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1\}$, while for the ABEES we use $k_g \in \{10, 15, 20\}$ and $k_l \in \{10, 15, 20, 25\}$, and $d \in \{\lceil|\Omega|\cdot 0.2\rceil, \lceil|\Omega|\cdot 0.3\rceil, \lceil|\Omega|\cdot 0.4\rceil, \lceil|\Omega|\cdot 0.5\rceil\}$ while $\delta = 0.0001$.

For what concern the NP method, we set up $k^{max} \in \{1, 5, 10, 15, 20, 30\}$. The proposed values may appear small, but the search is time consuming and the identification of each singleton may require the evaluation of thousands feasible points. Thus, we tested the method with $n_1 \in \{1, 5, 10, 15\}$ and with $N \in \{5, 10, 15, 20, 25\}$.

The tests have been conducted on a sub-set of the instances, and the number of tests (obtained as combination of the above parameters) have not been included for space grounds. Therefore, here we report the values that proved to be more effective (considering both performance of speed and solution quality) and that we have used in all the results provided in the following.

The η^{max} value has been fixed to 3000. For the RBEES, the probability p has been set-up to 0.4, while for the ABEES we fixed $k_g = 15$, $k_l = 20$ and $d = \lceil|\Omega|\cdot 0.2\rceil$. The k^{max} parameter within the NP method has been fixed to 10 and the parameters n_1 and N respectively to 1 and 5. The fact that the optimal value of n_1 is equal to 1 do not astonished the authors and it reflects some suspicious expressed during the study of the NP method proposed in Section 2.4 (our belief is that is a nonsense to take n_1 sets of N uniformly sampled points from a region for the evaluation of the performance of the region; at most is sufficient to produce one set of $n_1 \cdot N$ points).

3.5.2 Deterministic Optimization

In Table 3.3 is reported the solution found by the MIP solver (CPLEX), an the three metaheuristics (RBEES, ABEES and NP). The Table 3.3 reports for each instance the best solution found and the time in seconds spent during the search process. The best solution found is the best makespan (MS) found in a maximum time period of 15.000 seconds ($\approx$ 4 hours), therefore, the solution provided may not be the optimal. Every time the search process of the chosen methodology (i.e. the CPLEX solver or a metaheuristics) reaches the time-limit, then the search process is aborted: in this case, the time spent during the search process is denoted as *abort time limit* (ATL).

The metaheuristics RBEES, ABEES and NP have been executed within the same time limit of CPLEX, but with additional stopping criterions.

	CPLEX		RBEES		ABEES		NP	
Instance	*MS*	*Time*	*MS*	*Time*	*MS*	*Time*	*MS*	*Time*
1	419.78	307.8	421.44	0.3	421.44	0.3	423.11	0.5
2	347.11	4871.0	347.11	1.1	347.11	1.3	348.22	2.3
3	398.78	5170.8	398.78	0.6	398.78	0.5	398.78	0.8
4	602.91	*ATL*	608.78	16.1	608.78	32.0	610.44	71.5
5	563.90	*ATL*	564.56	4.2	569.67	4.8	568.11	18.2
6	594.00	*ATL*	598.22	11.5	599.33	24.5	596.33	34.3
7	527.23	*ATL*	584.89	22.3	541.33	43.8	537.33	109.2
8	638.56	*ATL*	644.11	39.9	665.67	58.8	659.78	251.7
9	646.22	*ATL*	657.22	38.1	673.00	63.8	658.67	288.1
10	627.77	*ATL*	627.33	181.7	648.44	192.6	639.44	683.7
11	722.55	*ATL*	722.55	140.3	724.44	179.8	733.00	588.5
12	690.99	*ATL*	690.99	168.0	698.44	241.2	692.11	2034.7
13	343.67	*ATL*	343.67	2735.2	343.78	2829.3	379.50	8655.3
Average	547.96	12334.6	554.62	258.4	555.75	283.8	557.29	979.9

Table 3.3: Comparison among CPLEX and the metaheuristics RBEES, ABEES and NP.

Results shown in Table 3.3 clearly demonstrate that the three metaheuristic approaches are well capable to select a good solution (sometimes the optimum) for most instances, even the most complicated. As reported in Table 3.4, the most effective method is the RBEES, but the performance of the ABEES method are surely comparable with those of the RBEES. The previous table also shows that the worst average performance is at most the 1.7% far from the CPLEX optimum.

The average time required by the RBEES, ABEES and NP method for finding the proposed (sub)optimal values are respectively 4.3, 4.7 and 16.3 minutes. This is not very much compared with the 205.6 minutes on average required by CPLEX. Obviously, the execution time is not the faster ever developed for the QCSP: but, this is quite normal considering that this search schemes are different by the others proposed in literature. In fact, these schemes are based on two time-consuming steps: *i*) the *generation of a feasible schedule* and, *ii*) the *evaluation of a schedule*. The other methodologies usually produce a feasible schedule whose makespan has been evaluated during the construction. Moreover, generally these methods exploit such a mathematical structure for the generation of a solution. However, the proposed metaheuristics are designed to evaluate a schedule under uncertainty, while other deterministic and faster methodologies are not capable to produce any results.

	RBEES	ABEES	NP
Instance	$\Delta\%_{MS}$	$\Delta\%_{MS}$	$\Delta\%_{MS}$
1	0.40%	0.40%	0.79%
2	0.00%	0.00%	0.32%
3	0.00%	0.00%	0.00%
4	0.97%	0.97%	1.25%
5	0.12%	1.02%	0.75%
6	0.71%	0.90%	0.39%
7	10.94%	2.67%	1.92%
8	0.87%	4.24%	3.32%
9	1.70%	4.14%	1.93%
10	0.00%	0.83%	1.86%
11	0.00%	0.26%	1.45%
12	0.00%	1.08%	0.16%

	RBEES	ABEES	NP
Instance	$\Delta\%_{MS}$	$\Delta\%_{MS}$	$\Delta\%_{MS}$
13	0.00%	0.03%	10.43%
Average	1.22%	1.42%	1.70%

Table 3.4: Comparison between the solution provided by CPLEX and the solutions provided by the RBEES, ABEES and NP method considering deterministic travel times and task processing time.

3.5.3 Simulation-based Optimization with non-deterministic times

Numerical experiments discussed in this section use a simplified simulation model of that described by the queuing network in Figure 3.1. Specifically, we have short-circuited both the "SC waiting line on quay" and the "TEUs waiting line under crane" with the purpose of isolating and highlighting the random effects of process discharge/loading times upon the schedules and, therefore, on the makespan. Therefore, we may still use the procedure for the evaluation of the makespan proposed in Section 3.4.1.

This Section aims to show how deterministic solutions found neither with MIP solvers and deterministic heuristic approaches are of such practical usefulness in a short term operative context. In fact, in the following the same instances are evaluated using the identical approaches and methodologies proposed in the previous Sections. Nonetheless, in the first stage of experiments, we use the RBEES(E), ABEES(E) and NP method for searching the best solution by using the procedure for the makespan evaluation proposed in Section 3.4.1, where the travel speed s_k for each crane $k \in K$ is a variate get by a random variable distributed as an exponential distribution with rate $\lambda_k = 45$ meters per minute. In the first stage we suppose that the discharge/loading time is still deterministic (therefore, the processing time p_i for the i-th task in Ω is consequently deterministic).

In the second stage, also the task processing time is assumed to be aleatory: in

particular, let x_i be the processing time for the task $i \in \Omega$, then x_i is a variate from an exponentially distributed random variable X_i with rate $\mu_i = 1/E[X_i]$, and $E[X_i] = p_i$.

Tests are performed using the same configurations adopted in the Section 3.5.2. Moreover, the base number of simulation runs for each evaluated solution has been set to 10.

Table 3.5 reports data on the 13 instances solved using an exponentially distributed travel speed for the RTGCs. Then, in Table 3.6 are given the results by the same tests conduced using exponentially distributed times for both travel time and task processing time. Successively, Table 3.7 gives the results of the simulation of the optimal schedules found by the ILOG's CPLEX solver using *i*) stochastic travel times and deterministic processing time, and *ii*) stochastic travel and processing times. The Table also reports the proportional increase/decrease on the makespan found by CPLEX using deterministic times.

	RBEES		ABEES		NP	
Instance	*MS*	*Time*	*MS*	*Time*	*MS*	*Time*
1	248.34	0.7	237.20	0.9	256.22	1.3
2	307.66	1.2	310.02	2.6	297.90	2.7
3	358.34	0.7	343.60	1.5	345.50	0.9
4	244.23	24.0	316.45	23.6	312.38	78.2
5	383.64	3.7	287.53	5.2	274.33	17.0
6	353.17	18.6	363.49	19.6	391.61	48.3
7	321.50	21.4	361.34	22.5	373.49	129.1
8	565.51	61.1	560.87	68.9	470.48	462.0
9	390.95	39.5	408.17	63.5	508.06	553.2
10	340.62	205.6	392.34	169.5	461.07	1124.5
11	455.74	64.1	496.98	110.0	524.53	1377.3
12	403.07	113.5	444.29	160.3	501.65	1035.2
13	268.85	1709.6	237.63	1569.9	298.82	*ATL*
Average	357.05	174.1	366.15	170.6	385.85	1549.0

Table 3.5: Comparison among the metaheuristics RBEES, ABEES and NP using a non-deterministic QCs speed.

	RBEES		ABEES		NP	
Instance	*MS*	*Time*	*MS*	*Time*	*MS*	*Time*
1	275.67	0.9	242.31	1.1	273.96	0.8
2	236.97	2.9	236.97	2.1	291.30	2.1
3	293.47	0.8	294.01	0.9	303.60	2.0
4	413.12	19.0	410.00	32.7	417.68	70.4
5	339.98	6.1	460.60	5.7	486.18	23.1
6	443.93	11.8	472.12	15.9	544.27	55.4
7	342.05	27.2	354.25	32.8	287.24	119.7
8	607.81	38.4	566.48	99.4	416.55	263.4
9	457.75	47.6	417.20	79.7	502.50	267.7
10	476.21	195.0	393.63	235.9	474.35	883.0
11	450.24	60.0	607.85	108.6	543.25	952.5
12	415.63	124.4	539.93	242.6	492.05	1331.5
13	293.04	1351.6	254.26	1369.9	313.76	*ATL*
Average	388.14	145.0	403.82	171.3	411.28	1459.4

Table 3.6: Comparison among the metaheuristics RBEES, ABEES and NP using a non-deterministic QCs speed and task processing time.

	CASE1	CASE2		CASE3	
Instance	*MS*	*MS*	$\Delta\%_{MS}$	*MS*	$\Delta\%_{MS}$
1	419.78	419.71	-0.02%	460.70	8.88%
2	347.11	346.72	-0.11%	361.87	4.08%
3	398.78	401.01	0.56%	548.14	27.25%
4	602.91	609.04	1.01%	752.93	19.92%
5	563.90	564.98	0.19%	685.25	17.71%
6	594.00	595.10	0.18%	782.87	24.13%
7	527.23	530.91	0.69%	40.77	28.83%
8	638.56	644.06	0.85%	854.83	25.30%
9	646.22	649.61	0.52%	1022.32	36.79%

	CASE1	CASE2		CASE3	
Instance	*MS*	*MS*	$\Delta\%_{MS}$	*MS*	$\Delta\%_{MS}$
10	627.77	636.25	1.33%	905.45	30.67%
11	722.55	724.45	0.26%	1059.47	31.80%
12	690.99	694.06	0.44%	1078.01	35.90%
13	343.67	345.80	0.62%	446.93	23.10%
Average	547.96	550.90	0.50%	746.12	24.18%

Table 3.7: Evaluation of the optimal schedule found by CPLEX by means of *Case* 1) deterministic times; *Case* 2) exponentially distributed RTGC travel speed and deterministic task processing time; *Case* 2) both RTGC travel speed and task processing time distributed by an exponential distribution.

The results of this first stage demonstrate that simulating the "optimal" solutions provided by CPLEX the estimated overall completion time is substantially identical than the deterministic value provided by CPLEX (in particular, the average value over all the 13 instances is higher than the 0.5%, as shown in Table 3.7).

The first stage introduces a little of randomness in the QCSP, but a substantial difference may be found between the best solution provided by the metaheuristics and the supposed optimal solution provided by CPLEX within the time-limit and successively simulated. On the real instance, the three methods need a lot of time to perform the search, and in particular, the NP has been stopped at the time-limit. This fact can be clarified considering that the instance 13 has a complex feasible region, therefore the partition of this region may require many additional estimates in order to identify the best subregion.

As reported in Table 3.5, the performance of the three metaheuristics compared with the performance of the CPLEX solution is globally better. As shown in Table 3.8, the average performance of the three algorithms adopting a simulation-based approach is globally better than using the deterministic-approach provided by CPLEX. In fact, the average value over all the 13 instances is lower than the 35.2% in the best case (i.e., using the RBEES(E) method) and by the 30.0% in the worst case (i.e., by using the NP method). The performance of the ABEES(E)

method are comparable with that of the RBEES(E).

This result shows clearly that the use of simulation for the identification of the best solution of a problem related to a system that operates under uncertainty is necessary, despite, as stated above, the first stage introduces a little of randomness. This is true because the RTGQs cover small distances when moving from a bay to another and this time is generally small with respect to the time required to process a task. Moreover, the results changes in a meaningful way if the travel speed (with directly influences the time required to cover the distance between two bays) reflect the possibility of such delay due to a crane breakdown.

	RBEES(E)	ABEES(E)	NP
Instance	$\Delta\%_{MS}$	$\Delta\%_{MS}$	$\Delta\%_{MS}$
1	-40.83%	-43.48%	-38.95%
2	-11.26%	-10.58%	-14.08%
3	-10.64%	-14.32%	-13.84%
4	-59.90%	-48.04%	-48.71%
5	-32.10%	-49.11%	-51.44%
6	-40.65%	-38.92%	-34.19%
7	-39.44%	-31.94%	-29.65%
8	-12.20%	-12.92%	-26.95%
9	-39.82%	-37.17%	-21.79%
10	-46.46%	-38.34%	-27.53%
11	-37.09%	-31.40%	-27.60%
12	-41.93%	-35.99%	-27.72%
13	-22.25%	-31.28%	-13.59%
Average	-35.19%	-33.54%	-29.96%

Table 3.8: Comparison between the solution provided by CPLEX (and successively simulated) and the solutions provided by the RBEES(E), ABEES(E) and NP method considering exponentially distributed RTGC travel speed and deterministic task processing time.

The second stage introduces another source of randomness in the QCSP. More

specifically, in the previous stage, the crane speed was distributed with an exponential time. Here, also the task processing time is exponentially distributed. The effects of this change are tremendous on the performance of the solutions proposed by CPLEX. In fact, simulating the "optimal" solutions provided by CPLEX, the estimated makespan is noticeably different than the deterministic value provided by CPLEX (in particular, the average value over all the 13 instances is the 24.2% far from the deterministic value, as shown in Table 3.7). In this case, as reported in Table 3.6, the results provided by the three metaheuristics are better than the values found by CPLEX. In particular, the solutions provided by CPLEX are globally the 44.0-48.0% worst than the values provided using the three algorithms. The results still confirm the trend of the performance of the three methods identified into the two cases (deterministic and with exponential travel times). The RBEES(E) method results to be the best search method, despite its search schema is quite simple. The performance of the ABEES(E) are comparable of that provided by the RBEES(E) method. The NP method, probably, require too much computational effort in a simulation-based approach, and the use of the BEES framework appear to be more appropriated.

	RBEES(E)	ABEES(E)	NP
Instance	$\Delta\%_{MS}$	$\Delta\%_{MS}$	$\Delta\%_{MS}$
1	-40.16%	-47.40%	-40.53%
2	-34.52%	-34.52%	-19.50%
3	-46.46%	-46.36%	-44.61%
4	-45.13%	-45.55%	-44.53%
5	-50.39%	-32.78%	-29.05%
6	-43.29%	-39.69%	-30.48%
7	-53.83%	-52.18%	-61.22%
8	-28.90%	-33.73%	-51.27%
9	-55.22%	-59.19%	-50.85%
10	-47.41%	-56.53%	-47.61%
11	-57.50%	-42.63%	-48.72%
12	-61.44%	-49.91%	-54.36%

	RBEES(E)	ABEES(E)	NP
Instance	$\Delta\%_{MS}$	$\Delta\%_{MS}$	$\Delta\%_{MS}$
13	-34.43%	-43.11%	-29.80%
Average	-47.98%	-45.88%	-44.88%

Table 3.9: Comparison between the solution provided by CPLEX (and successively simulated) and the solutions provided by the RBEES(E), ABEES(E) and NP method considering exponentially distributed both RTGC travel speed and task processing time.

Another interesting example may be pursued by using a hyper-exponentially distributed task processing time (e.g., the processing time may be higher for the effect of a *crane failure* or for the arrives of the next *shift work*). This example has been proposed by the authors in [44], with the purpose to finally demonstrate that SO is more suitable in a stochastic-dynamic environment than deterministic approaches. Finally, whenever the QCSP is contextualized in a larger dynamic-stochastic environment, e.g. within the queuing network proposed in Figure 3.1, other phenomena (e.g. cranes blocking and starvation) will surely must be taken into account. This complexity of relationship that made the system increasingly more realistic requires a discrete-event simulation model for the correct evaluation of the vessel overall completion time.

3.6 Conclusions

In this Chapter has been described the *quay crane scheduling problem* and the related logistical processes. In particular, a queuing network model has been proposed. The model is aimed to capture the key features of the logistic processes at hand, viewed as a dynamic, non deterministic process. Successively, a new mixed integer mathematical model has been proposed to be used for supporting allocation-scheduling decisions regarding quay cranes and vessel tasks to be discharged and/or loaded. Moreover, procedures for the development of a simulation-based optimization approach to the quay crane scheduling problem have been proposed. These procedures

have been integrated within the search schema of three recent and promising metaheuristics. The NP metaheuristic seems particularly promising in quickly providing cost effective solutions to the practical problem of determining the (sub)optimal assignment/schedule of quay cranes to vessel tasks. The RBEES proved to posses the best performance under uncertainty, while the ABEES is the best trade-off between solution quality and execution time. Currently, we are including more operational details in the simulation model to provide a finer representation of the discharge/loading process.

Appendix A

The Quay Crane Deployment Problem

The complex logistic process of vessel berthing followed by container discharge/loading, at maritime container terminals, is focused in this paper. Discrete-event simulation models are well capable of representing the entire process in a stochastic, dynamic environment. Hence, simulation results to be an effective planning and control tool for decision making and evaluation. The assignment of quay cranes to berthed vessels and their deployment along the berth represent crucial decisions that could be well supported by integer programming (IP) models. Usually, these models are used as standalone tools. Starting from a discrete-event simulator for the berth planning, previously developed for a real maritime container terminal, we propose two IP models that can be embodied within the simulator to verify whether or not the weekly plan of the berth schedule produced by the simulator itself is feasible with respect to the available quay cranes. If not, the manager would be asked to repeat the berth planning step by rerunning simulation. The goodness of the proposed IP formulations is established by a numerical comparison against a test case taken from literature.

A.1 Introduction

A maritime container terminal is a complex set of physical and human resources organized around a set of logistic processes. In a pure transhipment terminal, logistic processes are defined around the pure "store and forward" function of the terminal. This asks for the respect of high standards in the quality of service provided to shipping companies, otherwise, the terminal could lose some of these companies to competition. Thus the internal organization should be carefully optimized. Vice versa, in a different terminal devoted to an import/export function, possibly connected to a dry port [75], the logistic operations and infrastructures for optimally supporting inland/outland transportation by different modal choices should also be carefully considered [61] to improve the reasons of competitiveness.

In Chapter 1 we focused on the modeling of a high-level simulation model for a maritime container terminal. Her we goes deeper focusing the attention on the development of intelligent system for the optimal resource allocation.

Our preliminary consideration when addressing the modeling efforts referred to a marine container terminal, is that the major operational activities and resources should be managed by considering that they belong to multiple, interacting logistic processes (e.g. "vessel arrival" and "vessel discharge/loading" processes [91]) where some limited resources should be adequately shared. This fact is critical for a cost-effective management of the system and the choice of the modeling approach for performance evaluation and system optimization. Terminal competitiveness can be improved by optimizing the internal organization through the introduction of decision support systems in resource allocation and scheduling of logistic resources and operations [87, 86], with the objective of decreasing the operating costs and service times.

Several papers focus on IP models for specific processes in port logistics [60, 45, 16]. Besides, other papers based on simulation, such as [94, 43, 12] point out the opportunity of evaluating starvation and congestion phenomena occurring at those major resources on which both the terminal productivity and response time strongly depend.

In Section 1.6.1 at page 42 a queuing network model has been proposed. The

so called "inner model" described in that Section is equipped by a *manager component* (i.e. the "Operations Manager") whose tasks were to manage the container resource allocation. The most important resources that must be managed in a modern maritime container terminal are the berth and the quay cranes. The largest container terminal are equipped by a long berth and modern quay *rail-mounted gantry cranes* (RMGCs). Here we propose a methodology that can be developed within the "Operations Manager" to support the formulation of the so called *weekly plan* under the programmed flow of containership arrivals, provided that the statistical analysis of delays upon arrivals is continuously updated taking into account all the sources of uncertainty of the arrival process. In this context, the estimation of the vessel processing time (at berth) for discharge/loading operations is recognized as the second key point upon which the effectiveness of the whole planning process depends. The current release of the simulator asks the user to provide the so called "crane intensity", i.e. the average number of cranes that work, simultaneously, on the same vessel, as it is fixed by formal agreement with the shipping company to which the same vessel belongs. Hence, a decisional problem arises for the cranes manager because he should dynamically deploy the right number of RMGCs along the berth and assign these cranes to the vessels that succeed at the various berthing points, day by day, according to the weekly plan. The major constraints consist in respecting the vessels' due-time of departure and shifting the cranes (on rail) along the quay-side. The objective to pursue is that of employing the minimum number of quay cranes, while maximizing crane productivity.

In the language of Operations Research the above decisional problem may be referred as *the quay crane deployment problem* (QCDP) [39]. It is a complex assignment-scheduling problem that we tackle by two separated IP-based formulations. The first is focused on the assignment phase and produces the optimal number of cranes that must be assigned to each berthed vessel on the basis of a one-hour time-slot, under the guarantee that due-times of departure are respected. Work-shift are also considered in order to minimize the cost-per-use of the cranes. In the second IP formulation, the cumulative number of cranes returned by the first model is deployed over all the vessels previously berthed, each in their own position, in order to establish which cranes should service any given vessel on berth and

minimize the number of crane shifts between adjacent vessels.

The Appendix is organized as follows. In the following Section, the focus is on the QCDP and on the mathematical models applied within the two-phase approach. Finally, some computational results are presented.

This Appendix appeared in Legato, Gullì and Trunfio [41, 39].

A.2 The Quay Crane Deployment Problem

In Chapter 1 at page 35 we provided a description of the logistics and decision problems in a maritime container terminal, focusing the attention on a real terminal of pure transhipment: the container terminal in Gioia Tauro, Italy, which is situated along the major maritime routes from the far-east port sources in Asia to the ports of Northern Europe and other western destinations.

Here the focus is centered on the problem of allocating QCs mounted on rails at the incoming vessels considering a weekly plan that is known *a priori*.

An IP formulation of the *quay crane deployment problem* together the *berth allocation problem* (BAP) has been successfully discussed by Park and Kim [60]. In real life, the QCDP arises as follows.

The planning office of the terminal operating company constructs a weekly "berth schedule", which contains the berthing position and time window for each incoming vessel (this being the solution to the so-called *berth scheduling problem*). A time window shows the expected time of berthing and un-berthing for a single vessel; time windows are constructed using the ETA (Expected Time of Arrival) and PTD (Promised Time of Departure) of each vessel (a penalty cost must be sustained if the departure of a vessel occurs later than its previously committed PTD). Figure A.1 shows an example of a berth schedule, where the berth time and space are partitioned into 22×24 grid squares (24 one-hour time-slots).

The berth schedule is used to assign the RMG quay cranes to the incoming vessels on a daily basis. The double goal is to *i*) minimize the number of quay cranes to be employed and *ii*) maximize their utilization, under the constraint of completing the discharge/loading operations, for each vessel, within the related expected time

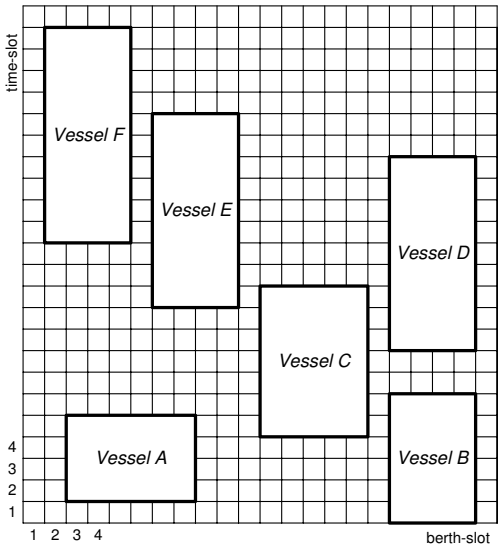


Figure A.1: An example of berth schedule presented by Park and Kim.

of un-berthing.

The QCDP is solved under the following assumptions.

1. Each vessel has a time window; the lower bound of the time window is the vessel's expected time of berthing, while the upper bound is the vessel's expected time of un-berthing.

2. Each vessel has a total number of TEUs to be handled within its time window: this number is related to the container discharge/loading moves required by the vessel.

3. Each vessel has a maximum and minimum number of cranes that can and must be assigned when operations starts. The maximum number of cranes that can be simultaneously assigned to a vessel is limited by vessel length. Vice versa, the minimum number of cranes to be assigned (usually for mother vessels) depends on the contract terms between the terminal operating company and the vessel's shipping company. By default, when operations start on a vessel, mostly all of the time-slots related to that vessel receive one crane.

4. Quay cranes are RMGCs, so non-crossing constraints must be guaranteed. Furthermore, cranes are never unavailable.

The solution approach we propose is decomposed into two phases, as shown in Figure A.2.

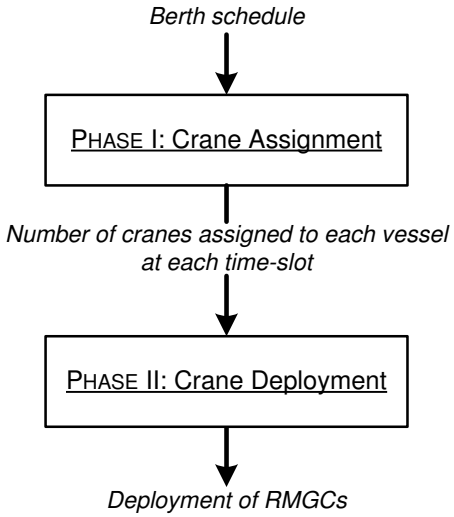


Figure A.2: The schema of the two-phase approach to the QCDP.

In the first phase (*crane assignment phase*), we solve an IP mathematical model using ILOG's CPLEX Solver 10.0 (tests were run on a 2.0 GHz Intel Pentium M computer with 2GB of memory) to identify the optimal number of cranes that must be assigned to each vessel at each time-slot considering the current workshift. Thus, the model is able to identify exactly when the discharge/loading operations start and end within the vessel's time-window. In literature, this problem is known as the *quay crane assignment problem* (QCAP) and it is usually studied together with the BAP as in [52].

In the second phase, another IP model is used to deploy the cranes along the quay, with the aim of matching the previously identified vessel-crane assignment in order to *i*) respect the non-crossing constraints, and *ii*)minimize the number of crane shifts from one vessel to another.

With respect to the two phases, the first IP model is the well-known QCAP, while the second is called the *quay crane deployment problem*; nevertheless, one

could give a mathematical model which combines both of the previous IP models and still refers to a QCDP.

A.2.1 A IP Model for the Crane Assignment Phase

The following notations will be used for the formulation of the QCAP. Let $\Omega = \{1, \ldots, \omega\}$ be the set of ω vessels, $T = \{1, \ldots, n\}$ be the set of n time-slots, and $K = \{1, \ldots, q\}$ be the set of q quay cranes (that moves on track).

Let also $\Phi = \{1, \ldots, m\}$ be a set of m work-shift on the time horizon, where the work-shift Φ_j is composed by $n_j - n_{j-1}$ time-slots. More precisely, the set of m work-shifts must be a partition of T, so that $\Phi_1 = \{1, 2, \ldots, n_1\}$, $\Phi_2 = \{n_1 + 1, \ldots, n_2\}$, $\ldots$, $\Phi_m = \{n_{m-1} + 1, \ldots, n\}$.

Let s_k be the service rate for the crane $k \in K$, expressed in TEUs per time-slot. Let m_i be the number of moves for the vessel $i \in \Omega$.

We also define for each vessel $i \in \Omega$: etb_i as the berthing time for vessel i, i.e. the time-slot starting from which vessel i is ready for the first lift, where $1 \leqslant etb_i \leqslant n - etu_i + 1$; etu_i the un-berthing time of vessel i; thus it is the last time-slot during which vessel i is available for operations, where $etb_i \leqslant etu_i \leqslant n$; min_i the minimum number of cranes that must be assigned to vessel i when operation starts; max_i the maximum number of cranes that can be assigned to vessel i. Moreover, for notational convenience, for each vessel $i \in \Omega$ we define the following sets of time-windows $\Upsilon_i = \{etb_i, \ldots, etu_i\}$, $\Upsilon_i^+ = \Upsilon_i \setminus \{etb_i\}$, $\Upsilon_i^- = \Upsilon_i \setminus \{etu_i\}$, and $\Upsilon = \bigcup_{i \in \Omega} \Upsilon_i$.

We introduce the following decisional variables:

- q_j is the maximum number of cranes used to perform discharging/loading operations on the vessels within the work-shift $j \in \Phi$.
- θ_{it}^k, for each vessel $i \in \Omega$, time-slot $t \in \Upsilon_i$ and RMGC $k \in K$, is equal to 1 if crane k works on vessel i at time-slot t, 0 otherwise.
- ϕ_{it}, for each vessel $i \in \Omega$, time-slot $t \in \Upsilon_i$, is equal to 1 if vessel i is processed at time-slot t, 0 otherwise.
- γ_{it}, for each vessel $i \in \Omega$, time-slot in $t \in \Upsilon_i$, is equal to 1 if operations for vessel i have been started at time-slot t, 0 otherwise.

- η_{it}, for each vessel $i \in \Omega$, time-slot $t \in \Upsilon_i$, is equal to 1 if operations for vessel i have not been completed at time-slot t, 0 otherwise.
- ρ_{it}, for each vessel $i \in \Omega$, time-slot $t \in \Upsilon_i^+$, is the difference between the number of cranes assigned at time-slot t and those assigned at the previous time-slot $(t-1)$.

The QCAP can be formulated as follows:

$$minimize \sum_{j \in \Phi} c_1 \cdot q_j + \sum_{i \in \Omega} \sum_{t \in \Upsilon_i} c_2 \cdot \phi_{it} + \sum_{i \in \Omega} \sum_{t \in \Upsilon_i^+} c_3 \cdot |\rho_{it}| \tag{A.1a}$$

subject to

$$\gamma_{it} \leqslant \gamma_{i\,(t+1)} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i^- \tag{A.1b}$$

$$\eta_{it} \leqslant \eta_{i\,(t+1)} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i^- \tag{A.1c}$$

$$\gamma_{it} + \eta_{it} = \phi_{it} + 1 \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i \tag{A.1d}$$

$$\sum_{k \in K} \sum_{t \in \Upsilon_i^-} s_k \theta_{it}^k = m_i \qquad \forall\, i \in \Omega \tag{A.1e}$$

$$\sum_{k \in K} \theta_{it}^k - \sum_{k \in K} \theta_{i(t-1)}^k = \rho_{it} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i^+ \tag{A.1f}$$

$$\sum_{i \in \Omega | t \in \Upsilon_i} \theta_{it}^k \leqslant 1 \qquad \forall\, t \in \Upsilon, \forall\, k \in K \tag{A.1g}$$

$$\sum_{k \in K} \theta_{it}^k \leqslant max_i \cdot \phi_{it} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i \tag{A.1h}$$

$$\sum_{k \in K} \theta_{it}^k \geqslant min_i \cdot \phi_{it} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i \tag{A.1i}$$

$$\sum_{k \in K} \sum_{i \in \Omega | t \in \Upsilon_i} \theta_{it}^k \leqslant q_j \qquad \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.1j}$$

$$q_j \leqslant q \qquad \forall\, j \in \Phi \tag{A.1k}$$

$$q_j \in \mathbb{N} \qquad \forall\, j \in \Phi \tag{A.1l}$$

$$\rho_{it} \in \mathbb{N} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon^+ \tag{A.1m}$$

$$\phi_{ij}, \gamma_{ij}, \eta_{ij} \in \{0,1\} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i \tag{A.1n}$$

$$\theta_{ij}^k \in \{0,1\} \qquad \forall\, i \in \Omega, \forall\, t \in \Upsilon_i, \forall\, k \in K \tag{A.1o}$$

We now explain the IP model (A.1).

The objective function (A.1a) aims to the minimization of *i*) the number of quay cranes employed in each work-shift of the planning horizon, *ii*) the overall number of time-slots required to perform vessel discharge/loading operations, and *iii*) the crane back and forth movements (implicitly accounted for by the $|\rho|$ factor, to be linearized as usual).

Clearly, function (A.1a) has been homogenized by introducing the cost multiplier c_1, c_2 and c_3, where: *i*) c_1 is the cost due to the activation of a crane during a work-shift (in fact, due to national contractual agreements, the labor force cannot be adopted whenever it is needed, but it must be employed within a known work-shift); *ii*) c_2 is the cost related to the processing time of a vessel, that is expressed in function of the overall number of time-slots on which the operations where performed with continuity; *iii*) c_3 is the cost due to the movement of a crane along the quay.

Constraints (A.1b)–(A.1d) ensure that for every vessel, once discharge/loading operations start, the operations must be performed without interruption until they are completed (the vessel operations cannot be preempted). Constraints (A.1e) specify that for each vessel, the discharge/loading operations must be executed and completed within the vessel time-window. Constraints (A.1f) evaluate the value of variables ρ_{it}. Constraints (A.1g) guarantee that every crane can be assigned to only one vessel at each time-slot. Constraints (A.1h) and (A.1i) ensure that the number of cranes assigned to a vessel during its operations time is between a *minimum* (i.e., due to contractual agreement) and a *maximum* (i.e., due to the vessel length). Constraints (A.1j) guarantee that, for each time-slot t, the number of assigned cranes result not greater than the number of available cranes. Constraints (A.1k) ensures that the number of cranes activated within the work-shift $j \in \Phi$ do not exceed the total number of quay cranes q. Constraints (A.1l)–(A.1o) are the basic constraints on the decision variables.

The IP model defined above is a refinement of the QCAP developed by Legato, Gullì and Trunfio [41, 39].

A.2.2 An IP Model for the Crane Deployment Phase

As stated before, the solution of the QCAP provides the number of cranes that must be assigned in order to complete the operations in time. This data is used in the following to deploy, for each time-slot, the RMG quay cranes. The deployment follows the criteria of non-crossing cranes and crane shifting reduction between vessels during different time-slots.

In the following we introduce the notations that will be used for the formulation of the quay crane deployment problem (QCDP).

Let Ω be the set of ω vessels, T be the set of n time-slots,Φ be the set of m work-shift and Φ^j, is the set of time-slot into the work-shift $j \in \Phi$, as introduced into the previous section. Let $K = \{q_1, \ldots, q_m\}$ be the set of rail-mounted quay cranes available over the m work-shift. Note that q_j is the minimum number of RMGCs necessary to perform discharging and loading operations for all the vessels in Ω into the work-shift $j \in \Phi^j$, with $q_j \leqslant q$. Let also $K^j = \{1, 2, \ldots, q_j\}$ be the cranes available within the work-shift $j \in \Phi^j$.

Let also: ac_t be the number of cranes assigned to the vessels berthed at time-slot t; ac_{it} be the number of cranes assigned to vessel i at time-slot t; cb_{it} be the number of cranes assigned to the vessels berthed at time-slot t before (from the left-side of the berth) vessel i; ca_{it} be the number of cranes assigned to the vessels berthed at time-slot t after (from the left-side of the berth) vessel i; w_i be the time-window of vessel i; this time-window is computed from the first time-slot in which discharge/loading operations start to the time-slot in which operations end. Note that these parameters are computed by the output from the first phase.

Moreover, the following decisional variables are introduced:

- ϕ_{it}^k, for each vessel $i \in \Omega$, work-shift $j \in \Phi$, time-slot $t \in \Phi^j$ and RMGC $k \in K^j$, is equal to 1 if crane k is assigned to vessel i at time-slot t, 0 otherwise.

- γ_{it}^k, for each vessel $i \in \Omega$, work-shift $j \in \Phi$, time-slot $t \in \Phi^j$ and RMGC $k \in K^j$, is equal to 1 if crane k is after (or is itself) the left-most crane assigned to vessel i at time-slot t.

- η_{it}^c, for each vessel $i \in \Omega$, work-shift $j \in \Phi$, time-slot $t \in \Phi^j$ and RMGC

$k \in K^j$, is equal to 1 if crane k is before (or is itself) the right-most crane assigned to vessel i at time-slot t.

- f_{it}, for each vessel $i \in \Omega$, time-slot $t \in T$, is the left-most crane assigned to vessel i at time-slot t.
- l_{it}, for each vessel $i \in \Omega$, time-slot $t \in T$, is the right-most crane assigned to vessel i at time-slot t.
- sf_i, for each vessel $i \in \Omega$, it is the sum over time of all the left-most crane indexes of vessel i.
- sl_i, for each vessel $i \in \Omega$, it is the sum over time of all the right-most crane indexes of vessel i.

Thus, the QCDP can be formulated as follows:

$$minimize \sum_{j \in \Phi} \sum_{t \in \Phi^j} \sum_{i \in \Omega} (q_j - ac_t) \cdot \left(\left| \frac{sf_i}{w_i} - f_{it} \right| + \left| \frac{sl_i}{w_i} - l_{it} \right| \right) \tag{A.2a}$$

subject to

$$\sum_{k \in K^j} \phi_{it}^k = ac_{it} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.2b}$$

$$\gamma_{it}^k \leqslant \gamma_{it}^{k+1} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j, \forall\, k \in K^j, k \neq q_j \tag{A.2c}$$

$$\eta_{it}^k \geqslant \eta_{it}^{k+1} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j, \forall\, k \in K^j, k \neq q_j \tag{A.2d}$$

$$\gamma_{it}^k + \eta_{it}^k = \phi_{it}^k + 1 \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j, \forall\, k \in K^j \tag{A.2e}$$

$$\sum_{k \in K^j} \gamma_{it}^k \leqslant q_j - cb_{it} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.2f}$$

$$\sum_{k \in K^j} \eta_{it}^k \leqslant q_j - ca_{it} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.2g}$$

$$\sum_{k \in K^j} \gamma_{it}^k + f_{it} = q_j + 1 \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.2h}$$

$$\sum_{k \in K^j} \eta_{it}^k - l_{it} = 0 \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.2i}$$

$$\sum_{t \in \Phi^j} f_{it} - sf_i = (q_j + 1) \cdot w_i \quad \forall\, i \in \Omega, \forall\, j \in \Phi \tag{A.2j}$$

$$\sum_{t \in \Phi^j} l_{it} - sl_i = 0 \quad \forall\, i \in \Omega, \forall\, j \in \Phi \tag{A.2k}$$

$$\phi_{it}^k, \gamma_{it}^k, \eta_{it}^k \in \{0, 1\} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j, \forall\, k \in K^j \tag{A.2l}$$

$$f_{it}, l_{it}, sf_i, sl_i \geqslant 0, \in \mathbb{N} \quad \forall\, i \in \Omega, \forall\, j \in \Phi, \forall\, t \in \Phi^j \tag{A.2m}$$

In the IP mathematical model (A.2), we aim to identify the RMGCs that must process the berthed vessels at each time-slot, with the goal of minimizing the number of cranes shifting from one vessel to another (i.e., if possible, the model tries to assign to each vessel always the same quay cranes).

Constraints (A.2b) specify that exactly the desired number of quay cranes is assigned to every vessel at each time-slot. Constraints (A.2c)–(A.2e) ensure that RMGCs are assigned with respect to non-crossing constraints.

Constraints (A.2f) force the assignment of quay cranes to vessel i at time-slot t, once the preceding vessels along the berth have received the expected amount of quay cranes. Likewise, constraints (A.2g) ensure that crane assignment to vessel i at time-slot t is done coherently with the assignment of the considered cranes to the vessels that follows along the berth.

Constraints (A.2h) identify the first crane from the left-side of the quay (i.e., the *left-most crane*) that must perform discharge/loading operations on vessel i at time-slot t; otherwise, constraints (A.2i) assign the last crane from the left-side of the quay (i.e., the *right-most crane*) that must perform discharge/loading operations on the same vessel and at the same time-slot. Likewise, constraints (A.2j) and (A.2k) extend constraints (A.2h) and (A.2i), respectively, over all the time-slots that constitute the vessel time-window.

Constraints (A.2l)–(A.2m) are the basic constraints on the decision variables.

A.3 Numerical Experiments

Numerical results obtained by the exact solution of the two mathematical formulations proposed in previous section are compared against the results reported by Park and Kim [60]. Here, we consider only one work-shift, because of the size of the instance.

In Figure A.3 we repeat the optimal assignment reported in the paper by Park and Kim based on the berth schedule previously shown in Figure A.1. We assumed that for each crane $k \in K$, the service rate $sc_k = 30$ TEUs per time-slot, and the moves for the vessels are respectively equal to 240, 720, 750, 810, 780, 900 TEUs. The cost have been assumed to be $c_1 = n$, $c_2 = c_3 = 1$. For each vessel, ringed numbers depict the quay cranes assigned to each vessel. As it is possible to see, the QCDP solved by Park and Kim makes use of 9 cranes to complete all the operations in time.

In this case study, the minimum number of cranes that must be assigned to the vessels during operations is one, while the maximum number of cranes that can be assigned to a specific vessel is equal to the number of occupied berth-slots (each corresponding to 50 meters).

The first step of our approach produces the assignment depicted in Figure A.4(a). In this berth schedule, the optimal value of ac_{it}, for each couple of vessel i and time-slot t, is reported within each corresponding rectangle.

As it is easy to recognize, our mathematical model fills-in a berth schedule while minimizing the overall number of cranes that must be used to process all of the

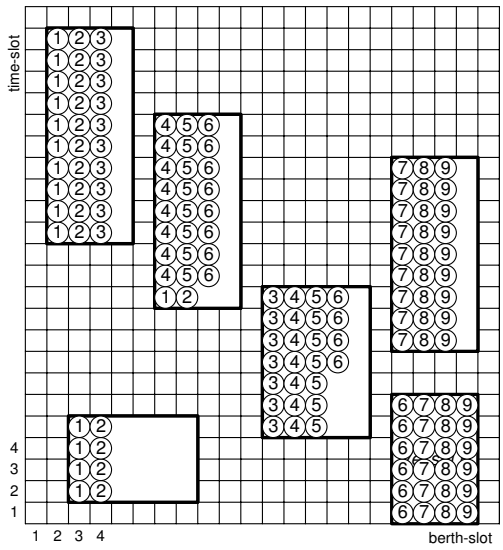
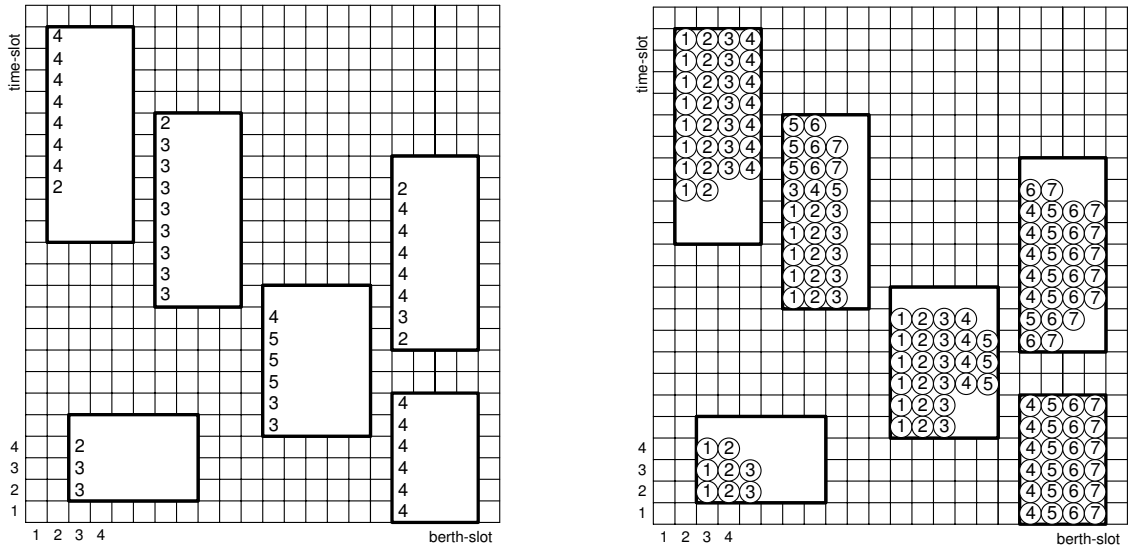


Figure A.3: The optimal deployment found by the Park and Kim methodology.



(a) Output from the first phase.

(b) Output from the second phase.

Figure A.4: The output of the proposed methodology.

vessels. In fact, the first phase of our approach produced an assignment of 7 quay cranes against the 9 quay cranes found by Park and Kim. Moreover, in our solution, once operations start, no operation discontinuity can occur for any vessel: this is

a primary contractual agreement requested to the terminal operating company by shipping companies, along with the minimum number of cranes to be assigned and the respect of the bounds on operation completion time.

As a result of the second phase of our methodology, we propose the quay crane deployment shown in Figure A.4(b).

The improvement obtained with the approach proposed in this paper is not only due to the minimization of the activated cranes. In fact, in three cases i) we obtained a reduction of the vessel overall completion time and ii) we improved the average crane utilization (*0.69 vs. 0.87*).

E better results may be achieved considering that a normal work-shift is composed by 6 hours. Therefore, the previous example mat have 4 work-shift, and the maximum number of cranes assigned to each work-shift will be 6 for the last work-shift and 7 for the others.

A.4 Conclusion and future development

In Chapter 1 we have reported on the possibility of improving the benefit of using a simulator when managing the logistic process of vessel berthing and discharge/loading at a maritime container terminal. Combining discrete-event simulation with integer programming models results in a very powerful tool, where the solution of assignment and scheduling problems plays a crucial role.

Here, we have proposed a practical solution to the problem of guaranteeing the respect of the planned time windows (berthing–un-berthing) but also pursuing the objective of minimizing quay crane idle time. The novelty is that we avoid handling a unique, unmanageable formulation and, furthermore, that for practical applications in real life, we may use a berth schedule produced by a simulator, i.e. in a more realistic modeling environment where operation delays and unpredictable events may occur. Thus we may integrate the IP models with their respective solution algorithms in a discrete-event simulator to support run-time crane assignment using a berth schedule. A metaheuristics based approach will be clearly pursued to develop solution algorithms for real instances; this will also enable simulation-based optimization features within the a discrete-event simulator. The proposed

methodology can also be used to practically approach deterministic problems.

Appendix B

Instances of the Quay Crane Scheduling Problem

Here we present the instances that have been used in Chapter 3 for the comparison with the solution provided using a commercial MIP solver (i.e, the ILOG's CPLEX) and the metaheuristic-based approach.

B.1 Instances

As defined in Section 3.3.1, let Ω be the set of tasks and K be the set of quay cranes. Let also B be the number of ship-bays for a specific QCSP's instance. Let B_i be the number of the bay of the task $i \in \Omega$, with $B_i \leqslant B$. The average operation time for a task has been randomly generated from a uniform distribution of $U\{1, 190\}$. Also, attributes of tasks were chosen randomly. The attributes are: *bay number* (Bi), *loading* (L) or *unloading* (U), *deck task* (D) or *hold task* (H) , *average processing time* (P) and *spacing among task's bay and the first bay* (S). Thus, we report the parameter for the ith task as $Ti(Bi,\ L$ or $U,\ D$ or $H,\ P,\ S)$.

We consider a distance of $\bar{d}$=25m between the center of adjacent couple of bays. An exception is considered for instance 13, that has been designed around a real vessel: in this case, an additional distance of 12.5m is placed between bays 2–3 and 6–7. We assumed that the average travel time of each quay crane is $\bar{s}$=45m/min. Therefore, the time t_{ij} required to cover the distance between two task $i, j \in \Omega$ is

computed as follows:

$$t_{ij} = \frac{\bar{d}}{\bar{s}} \cdot |B_i - B_j|$$

The set Φ of precedence relationship among tasks in the same bay and the set Ψ of non-simultaneity among tasks from adjacent bays are considered after the deployment of the tasks within the ship's bays. Obviously, is known that $\Phi \subseteq \Psi$, but the structure of the mathematical models proposed in Section 3.3 and the procedure used for the evaluation of the makespan (see Section 3.4.1), let us to compute the set of non-simultaneity constraints as $\Psi = \Psi \backslash \Phi$.

Instance 1

- $|\Omega| = 10$.
- $|K| = 2$.
- $B = 10$.
- *T1*(B1, L, D, 82, 0), *T2*(B2, U, D, 158, 25), *T3*(B2, U, H, 62, 25), *T4*(B4, U, H, 27, 75), *T5*(B5, L, D, 154, 100), *T6*(B6, U, D, 11, 125), *T7*(B7, L, D, 121, 150), *T8*(B7, L, H, 53, 150), *T9*(B9, L, H, 150, 200), *T10*(B10, U, H, 12, 225).
- $\Phi = \{(3, 2), (9, 8)\}$.
- $\Psi = \{(1, 2), (1, 3), (5, 6), (7, 8), (9, 10)\}$.

Instance 2

- $|\Omega| = 10$.
- $|K| = 2$.
- $B = 10$.
- *T1*(B1, U, H, 26, 0), *T2*(B3, U, D, 87, 50), *T3*(B4, U, D, 47, 75), *T4*(B4, L, D, 48, 75), *T5*(B4, L, H, 34, 75), *T6*(B5, U, H, 67, 100), *T7*(B5, L, H, 89, 100), *T8*(B5, L, D, 70, 100), *T9*(B7, U, D, 120, 150), *T10*(B9, U, D, 98, 200).

- $\Phi = \{(3,5), (5,4), (7,8), (8,6)\}$.
- $\Psi = \{(2,3), (2,4), (2,5), (3,6), (3,7), (3,8), (4,6), (4,7), (4,8), (5,6), (5,7), (5,8)\}$.

Instance 3

- $|\Omega| = 10$.
- $|K| = 2$.
- $B = 10$.
- *T1*(B1, U, D, 77, 0), *T2*(B1, L, H ,46, 0), *T3*(B2, L, D, 56, 25), *T4*(B4, L, H, 78, 75), *T5*(B5, U, H, 159, 100), *T6*(B5, L, D, 168, 100), *T7*(B6, U, H, 90, 125), *T8*(B8, U, D, 39, 175), *T9*(B8, L, H, 5, 175), *T10*(B10, L, H, 69, 225).
- $\Phi = \{(1,2), (6,5), (8,9)\}$.
- $\Psi = \{(1,3), (2,3), (4,5), (4,6), (5,7), (6,7)\}$.

Instance 4

- $|\Omega| = 15$.
- $|K| = 2$.
- $B = 10$.
- *T1*(B2, U, D, 23, 25), *T2*(B2, L, D, 89, 25), *T3*(B2, U, H, 38, 25), *T4*(B2, L, D, 103, 25), *T5*(B3, U, H, 11, 50), *T6*(B4, L, D, 90, 75), *T7*(B4, U, H 103, 75), *T8*(B5, U, H, 67, 100), *T9*(B6, L, D, 137, 125), *T10*(B7, U, D, 56, 150), *T11*(B7, U, H, 178, 150), *T12*(B9, U, D, 45, 200), *T13*(B10, U, D, 90, 225), *T14*(B10, U, H 45, 225), *T15*(B10, L, H, 120, 225).
- $\Phi = \{(1,3), (3,4), (4,2), (7,6), (10,11), (13,14), (14,15)\}$.
- $\Psi = \{(1,5), (2,5), (3,5), (4,5), (5,6), (5,7), (6,8), (7,8), (8,9), (9,10), (9,11), (12,13), (12,14), (12,15)\}$.

Instance 5

- $|\Omega| = 15$.
- $|K| = 2$.
- $B = 10$.
- *T1*(B1, U, H, 45, 0), *T2*(B2, L, D, 67, 25), *T3*(B2, U, H, 120, 25), *T4*(B3, U, D, 50, 50), *T5*(B3, L, H, 45, 50), *T6*(B4, U, D 45, 75), *T7*(B4, U, H, 120, 75), *T8*(B5, L, H, 90, 100), *T9*(B6, L, H, 120, 125), *T10*(B7, L, D, 30, 150), *T11*(B7, L, H, 180, 150), *T12*(B8, U, D, 34, 175), *T13*(B8, U, H, 67, 175), *T14*(B9, U, H, 99, 200), *T15*(B10, U, H, 5, 225).
- $\Phi = \{(3,2), (4,5), (6,7), (11,10), (12,13)\}$.
- $\Psi = \{(1,2), (1,3), (2,4), (2,5), (3,4), (3,5), (4,6), (4,7), (5,6), (5,7), (6,8), (7,8), (8,9), (9,10), (9,11), (10,12), (10,13), (11,12), (11,13), (12,13), (13,14), (14,15)\}$.

Instance 6

- $|\Omega| = 15$.
- $|K| = 2$.
- $B = 10$.
- *T1*(B1, L, D, 25, 0), *T2*(B1, U, H, 34, 0), *T3*(B2, U, D, 90, 25), *T4*(B2, U, H, 171, 25), *T5*(B3, U, H, 56, 50), *T6*(B3, L, H, 43, 50), *T7*(B4, L, D, 90, 75), *T8*(B4, U, H, 146, 75), *T9*(B6, L, H, 33, 125), *T10*(B7, L, D, 180, 150), *T11*(B8, L, D, 45, 175), *T12*(B8, U, H, 70, 175), *T13*(B8, U, D, 56, 175), *T14*(B10, L, D, 78, 225), *T15*(B10, U, H, 56, 225).
- $\Phi = \{(2,1), (3,4), (5,6), (8,7), (12,13), (13,11), (15,14)\}$.
- $\Psi = \{(1,3), (1,4), (2,3), (2,4), (3,5), (3,6), (4,5), (4,6), (5,7), (5,8), (6,7), (6,8), (9,10), (10,11), 10,12), (10,13)\}$.

Instance 7

- $|\Omega| = 20$.
- $|K| = 3$.
- $B = 20$.
- *T1*(B1, U, D, 23, 0), *T2*(B2, U, D, 145, 25), *T3*(B2, L, H, 34, 25), *T4*(B3, U, D, 22, 50), *T5*(B4, U, D, 32, 75), *T6*(B4, L, H, 167, 75), *T7*(B7, L, D, 121, 150), *T8*(B7, L, H, 53, 150), *T9*(B9, L, H, 120, 200), *T10*(B10, U, H, 89, 225), *T11*(B11, U, D, 76, 250), *T12*(B12, L, D, 134, 275), *T13*(B12, L, H, 89, 275), *T14*(B14, L, D, 45, 325), *T15*(B14, U, H, 5, 325), *T16*(B14, L, H, 111, 325), *T17*(B15, U, H, 67, 350), *T18*(B16, U, D, 87, 375), *T19*(B17, U, D, 92, 400), *T20*(B17, U, H, 28, 400).
- $\Phi = \{(2,3), (5,6), (8,7), (13,12), (15,16), (16,14), (19,20)\}$.
- $\Psi = \{(1,2), (1,3), (2,4), (3,4), (4,5), (4,6), (9,10), (10,11), (11,12), (11,13), (14,17), (15,17), (16,17), (17,18), (18,19), (18,20)\}$.

Instance 8

- $|\Omega| = 20$.
- $|K| = 3$.
- $B = 20$.
- *T1*(B1, L, H, 11, 0), *T2*(B4, L, D, 78, 75), *T3*(B4, U, H, 87, 75), *T4*(B4, L, H, 45, 75), *T5*(B5, U, H, 135, 100), *T6*(B7, L, D, 145, 150), *T7*(B9, L, H, 56, 200), *T8*(B10, U, D, 90, 225), *T9*(B10, L, D, 45, 225), *T10*(B10, U, H, 65, 225), *T11*(B10, L, H, 101, 225), *T12*(B13, U, D, 145, 300), *T13*(B14, L, H, 99, 325), *T14*(B16, U, H, 76, 375), *T15*(B16, L, H, 47, 375), *T16*(B17, L, D, 89, 400), *T17*(B19, U, D, 99, 450), *T18*(B19, U, H, 123, 450), *T19*(B19, L, H, 145, 450), *T20*(B20, U, H, 178, 475).
- $\Phi = \{(3,4), (4,2), (8,10), (10,11), (11,9), (14,15), (17,18), (18,19)\}$.

- Ψ = {(2,5), (3,5), (4,5), (7,8), (7,9), (7,10), (7,11), (13,14), (14,16), (15,16), (17,20), (18,20), (19,20)}.

Instance 9

- $|\Omega| = 20$.
- $|K| = 3$.
- $B = 20$.
- *T1*(B1, U, D, 65, 0), *T2*(B1, L, D, 45, 0), *T3*(B1, L, H, 10, 0), *T4*(B3, U, H, 68, 50), *T5*(B5, L, H, 78, 100), *T6*(B6, U, H, 90, 125), *T7*(B6, L, H, 67, 125), *T8*(B9, U, H, 174, 200), *T9*(B10, L, D, 167, 225), *T10*(B10, U, H, 145, 225), *T11*(B13, L, D, 77, 300), *T12*(B13, U, H, 87, 300), *T13*(B13, L, H, 145, 300), *T14*(B14, U, D, 34, 325), *T15*(B16, U, H, 170, 375), *T16*(B16, L, H, 7, 375), *T17*(B18, L, H, 156, 425), *T18*(B19, U, D, 78, 450), *T19*(B19, U, H, 99, 450), *T20*(B19, L, H, 135, 450).
- Φ = {(1,3), (3,2), (6,7), (10,9), (12,13), (13,11), (16,15), (18,19), (19,20)}.
- Ψ = {(5,6), (5,7), (8,9), (8,10), (11,4), (12,14), (13,14), (17,18), (17,19), (17,20)}.

Instance 10

- $|\Omega| = 25$.
- $|K| = 3$.
- $B = 20$.
- *T1*(B1, L, D, 10, 0), *T2*(B2, U, H, 89, 25), *T3*(B3, L, D, 78, 50), *T4*(B3, U, H, 14, 50), *T5*(B3, L, H, 67, 50), *T6*(B5, L, D, 87, 100), *T7*(B5, L, H, 78, 100), *T8*(B7, U, D, 66, 150), *T9*(B7, L, H, 145, 150), *T10*(B9, L, D, 67, 200), *T11*(B9, U, H, 45, 200), *T12*(B9, L, H, 98, 200), *T13*(B10, U, D, 123, 225), *T14*(B11, U, D, 139, 250), *T15*(B12, L, H, 22, 275), *T16*(B13, U, H, 13, 300),

T17(B14, L, D, 22, 325), *T18*(B14, U, H, 89, 325), *T19*(B15, L, H, 17, 350), *T20*(B16, L, D, 176, 375), *T21*(B17, U, H, 56, 400), *T22*(B19, U, D, 45, 450), *T23*(B19, U, H, 56, 450), *T24*(B19, L, H, 143, 450), *T25*(B20, U, H, 78, 475).

- $\Phi = \{(4,5), (5,3), (7,6), (8,9), (11,12), (12,10), (18,17), (22,23), (23,24)\}$.
- $\Psi = \{(2,3), (2,4), (2,5), (10,13), (11,13), (12,13), (13,14), (14,15), (15,16), (16,17), (16,18), (17,19), (18,19), (19,20), (20,21), (22,25), (23,25), (24,25)\}$.

Instance 11

- $|\Omega| = 25$.
- $|K| = 3$.
- $B = 20$.
- *T1*(B2, L, D, 67, 25), *T2*(B2, U , H, 120, 25), *T3*(B3, L, D, 45, 50), *T4*(B3, L, H, 98, 50), *T5*(B4, U, D, 56, 75), *T6*(B5, U, H, 120, 100), *T7*(B5, L, H, 88, 100), *T8*(B7, U, D, 87, 150), *T9*(B7, L, H, 123, 150), *T10*(B9, L, D, 56, 200), *T11*(B9, U, H, 87, 200), *T12*(B9, L, H, 120, 200), *T13*(B10, U, D, 125, 225), *T14*(B11, U, D, 167, 250), *T15*(B12, L, H, 7, 275), *T16*(B13, U, H, 45, 300), *T17*(B14, L, D, 3, 325), *T18*(B14, U, H, 34, 325), *T19*(B15, L, H, 25, 350), *T20*(B16, L, D, 140, 375), *T21*(B17, U, H, 98, 400), *T22*(B19, U, D, 39, 450), *T23*(B19, U, H, 120, 450), *T24*(B19, L, H, 69, 450), *T25*(B20, U, H, 156, 475).
- $\Phi = \{(2,1), (3,4), (6,7), (8,9), (11,12), (12,10), (18,17), (22,23), (23,24)\}$
- $\Psi = \{(1,3), (1,4), (2,3), (2,4), (2,4), (3,5), (4,5), (5,6), (5,7), (10,13), (11,13), (12,13), (13,14), (14,15), (15,16), (16,17), (16,18), (17,19), (18,19), (19,20), (20,21), (22,25), (23,25), (24,25)\}$

Instance 12

- $|\Omega| = 25$.

- $|K| = 3$.
- $B = 20$.
- *T1*(B1, U, H, 4, 0), *T2*(B2, U, D, 55, 25), *T3*(B3, L, D, 33, 50), *T4*(B4, U, D, 34, 75), *T5*(B4, U, H, 67, 75), *T6*(B4, L, H, 4, 75), *T7*(B5, U, H, 99, 100), *T8*(B5, L, H, 134, 100), *T9*(B7, L, D, 59, 150), *T10*(B7, L, H, 111, 150), *T11*(B9, U, H, 50, 200), *T12*(B9, L, H, 101, 200), *T13*(B10, L, H, 123, 225), *T14*(B11, U, D, 198, 250), *T15*(B12, U, H, 23, 275), *T16*(B12, L, H, 27, 275), *T17*(B14, L, D, 79, 325), *T18*(B14, U, H, 156, 325), *T19*(B15, L, H, 59, 350), *T20*(B16, L, D, 126, 375), *T21*(B16, U, H, 146, 375), *T22*(B18, U, D, 78, 425), *T23*(B18, U, H, 175, 425), *T24*(B19, U, D, 19, 450), *T25*(B19, L, D, 29, 450).
- Φ = {(4,5), (5,6), (7,8), (10,9), (11,12), (15,16), (18,17), (20,21), (22,23), (24,25)}.
- Ψ = {(1,2), (2,3), (3,4), (3,5), (3,6), (4,7), (4,8), (5,7), (5,8), (6,7), (6,8), (11,13), (12,13), (13,14), (14,15), (14,16), (17,19), (18,19), (19,20), (19,21), (22,24), (22,25), (23,24), (23,25)}.

Instance 13

- $|\Omega| = 24$.
- $|K| = 3$.
- $B = 8$.
- *T1*(B1, U, D, 45, 0), *T2*(B1, L, D, 33, 0), *T3*(B1, U, H, 52, 0), *T4*(B1, L, H, 24, 0), *T5*(B2, U, D, 53, 25), *T6*(B2, L, D, 6, 25), *T7*(B2, U, H, 51, 25), *T8*(B2, L, H, 38, 25), *T9*(B3, U, D, 35, 62.5), *T10*(B3, L, D, 12, 62.5), *T11*(B4, U, D, 38, 87.5), *T12*(B4, L, D, 11, 87.5), *T13*(B4, U, H, 72, 87.5), *T14*(B4, L, H, 40, 87.5), *T15*(B5, U, D, 43, 112.5), *T16*(B5, L, D, 11, 112.5), *T17*(B5, U, H, 44, 112.5), *T18*(B5, L, H, 64, 112.5), *T19*(B6, U, D, 1, 137.5), *T20*(B7, U, D, 87, 175), *T21*(B7, L, D, 10, 175), *T22*(B7, U, H, 38, 175), *T23*(B7, L, H, 34, 175), *T24*(B8, U, D, 82, 200).

- Φ = {(1,3), (3,4), (4,2), (5,7), (7,8), (8,6), (9,10), (11,13), (13,14), (14,12), (15,17), (17,18), (18,16), (20,22), (22, 23), (23,21)}.
- Ψ = {(1,5), (1,7), (3,5), (3,7), (2,5), (2,7), (4,5), (4,7), (1,6), (1,8), (3,6), (3,8), (2,6), (2,8), (4,6), (4,8), (9,11), (9,13), (10,11), (10,13), (9,12), (9,14), (10,12), (10,14), (11,15), (11,17), (11,16), (11,18), (13,15), (13,17), (13,16), (13,18), (12,15), (12,17), (12,16), (12,18), (14,15), (14,17), (14,16), (14,18), (15,19), (17,19), (16,19), (18,19), (20,24), (22,24), (21,24), (23,24)}.

Bibliography

[1] S. Andradóttir, *A review of simulation optimization techniques*, Proceedings of the 1998 Winter Simulation Conference (Washington, DC, USA), December 1998, pp. 151–158.

[2] ———, *Accelerating the convergence of random search methods for discrete stochastic optimization*, ACM Transactions on Modeling and Computer Simulation **9** (1999), no. 4, 349–380.

[3] ———, *Simulation optimization with countably infinite feasible regions: Efficiency and convergence*, ACM Transactions on Modeling and Computer Simulation **16** (2006), no. 4, 357–374.

[4] J. April, F. Glover, J.P. Kelly, and M. Laguna, *Simulation/optimization using "real-world" applications*, Proceedings of the 2001 Winter Simulation Conference (New Orleans, Lousiana, USA), December 2001, pp. 71–78.

[5] ———, *Practical introduction to simulation optimization*, Proceedings of the 2003 Winter Simulation Conference (New Orleans, Louisiana, USA), December 2003, pp. 71–78.

[6] J. Banks, B.L. Carson, J.S. Nelson, and D.M. Nicol, *Discrete event systems simulation*, 3rd ed., Prentice Hall, Englewood Cliffs, New Jersey, 2000.

[7] M.R.P. Barretto, L. Chwif, T. Eldabi, and R.J. Paul, *Simulation optimization with the linear move and exchange move optimization algorithm*, Proceedings of the 1999 Winter Simulation Conference (Phoenix, Arizona, USA), December 1999, pp. 806–811.

[8] J. Boesel, R.O. Bowden, Jr., F. Glover, J.P. Kelly, and E. Westwig, *Future of simulation optimization*, Proceedings of the 2001 Winter Simulation Conference (Arlington, Virginia, USA), December 2001, pp. 1466–1469.

[9] Allen Bradley, *Optquest for arena*, Rockwell Automation, 2007.

[10] P. Brucker, *Scheduling algorithms*, 5 ed., Springer, 2007.

[11] P. Canonaco, P. Legato, and R.M. Mazza, *An integrated simulation model for channel contention and berth management at a maritime container terminal*, Proceedings of the 21th European Conference on Modelling and Simulation (Prague, Czech Republic), June 2007, pp. 353–362.

[12] P. Canonaco, P. Legato, R.M. Mazza, and R. Musmanno, *A queuing network model for the management of berth crane operations*, Computers and Operations Research **35** (2008), 2432–2446.

[13] E. Chen, C.-H. andYücesan, Y. Yuan, H.-C. Chen, and L. Dai, *Computing budget allocation for simulation experiments with different system structures*, Proceedings of the 1998 Winter Simulation Conference (Washington, D.C., USA), December 1998, pp. 753–741.

[14] S.E. Chick, *Selecting the best system: A decision theoretic approach*, Proceedings of the 1997 Winter Simulation Conference (Atlanta, Georgia, USA), December 1997, pp. 326–333.

[15] J.-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia, *The service allocation problem at the gioia tauro maritime terminal*, European Journal of Operational Research **176** (2007), 1167–1184.

[16] J.-F. Cordeau, G. Laporte, P. Legato, and L. Moccia, *Models and tabu search heuristics for the berth allocation problem*, Transportation Science **39** (2005), no. 4, 526–538.

[17] B. Cota and R. Sargent, *Control flow graphs: A method of model representation for parallel discrete event simulation*, CASE Center Technical Report 9026, Syracuse University, 1990.

[18] ———, *A modification of the process interaction world view*, ACM Transactions on Modelling and Computer Simulation **2** (1992), no. 2, 109–129.

[19] C.F. Daganzo, *The crane scheduling problem*, Transport Research, Part B **23** (1989), no. 3, 159–175.

[20] O. Dahl and K. Nygaard, *Simula - an algol-based simulation language*, Communications of the ACM **9** (1966), no. 9, 349–395.

[21] E.J. Derrick, O. Balci, and R.E. Nance, *A comparison of selected conceptual frameworks for simulation modelling*, Proceedings of the 1989 Winter Simulation Conference (Blacksburg, Virginia, USA), December 1989, pp. 711–718.

[22] D.G. Fritz and R.G. Sargent, *An overview of hierarchical control flow graph models*, Proceedings of the 1995 Winter Simulation Conference (Arlington, Virginia, USA), December 1995, pp. 1347–1355.

[23] M.C. Fu, *Encyclopedia of operations research and management science*, 2nd ed., ch. Simulation Optimization, pp. 756–759, Kluwer Academic Publishers, Boston, 2000.

[24] ———, *Simulation optimization*, Proceedings of the 2001 Winter Simulation Conference (Arlington, Virginia, USA), December 2001, pp. 53–61.

[25] M.C. Fu, S. Andradóttir, J.S. Carson, F. Glover, C.R. Harrell, Y.C. Ho, J.P. Kelly, and S.M. Robinson, *Integrating optimization and simulation: Research and practice*, Proceedings of the 2000 Winter Simulation Conference (Orlando, Florida, USA), December 2000, pp. 610–616.

[26] M.C. Fu, F.W. Glover, and J. April, *Simulation optimization: A review, new developments, and applications*, Proceedings of the 2005 Winter Simulation Conference (Orlando, Florida, USA), December 2005, pp. 83–95.

[27] G. Ghiani, P. Legato, R. Musmanno, and F. Vocaturo, *Optimization via simulation: Solution concepts, algorithms, parallel computing strategies and commercial software*, International Scientific Journal of Computing **3** (2004), no. 3, 7–12.

[28] F. Glover, *Tabu search – part i*, ORSA Journal on Computing **1** (1989), no. 3, 190–206.

[29] ———, *Tabu search – part ii*, ORSA Journal on Computing **2** (1990), no. 1, 4–32.

[30] R. Horst and H. Tuy, *Global optimization: Deterministic approaches*, Springer-Verlag, 1990.

[31] K. Jensen, *Advances in petri nets 1990*, Lecture Notes in Computer Science, vol. 483, ch. Coloured Petri nets: A high level language for system design and analysis, pp. 342–416, Springer Berlin/Heidelberg, 1991.

[32] K.H. Kim and Y.M. Park, *A crane scheduling method for port container terminals*, European Journal of Operational Research **156** (2004), 752–768.

[33] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, *Optimization by simulated annealing*, Science, New Series **220** (1983), no. 4598.

[34] A.H. Land and A.G. Doing, *An automatic method of solving discrete programming problems*, Econometrica **28** (1960), no. 3, 497–520.

[35] A.M. Law and W.D. Kelton, *Simulation modelling and analysis*, 3rd ed., McGraw-Hill, 2000.

[36] A.M. Law and M.G. McComas, *Simulation-based optimization*, Proceedings of the 2002 Winter Simulation Conference (San Diego, California, USA), December 2002, pp. 41–44.

[37] P. L'Ecuyer, *Scaling, hierarchical modelling, and reuse in an object-orinted modelling and simulation systems*, Proceedings of the 1999 Winter Simulation Conference (Phoenix, Arizona, USA), December 1999, pp. 95–105.

[38] ———, *Software for uniform random number generation: distinguishing the good and the bad*, Proceedings of the 2001 Winter Simulation Conference, (Arlington, Virginia, USA), December 2001, pp. 95–105.

[39] P. Legato, D. Gullì, and R. Trunfio, *Assignment and deployment of quay cranes at a maritime container terminal*, Proceedings of the 11th International Workshop on Harbour, Maritime & Multimodal Logistics Modeling & Simulation (Amantea, Italy), September 2008, pp. 214–220.

[40] ———, *Modeling, simulation and optimization of logistics processes*, Proceedings of 20th European Modeling and Simulation Symposium (Simulation in Industry) (Amantea, Italy), September 2008, pp. 569–578.

[41] ———, *The quay crane deployment problem at a maritime container terminal*, Proceedings of the 22th European Conference on Modelling and Simulation (Nicosia, Cyprus), June 2008, pp. 53–59.

[42] P. Legato, D. Gullì, R. Trunfio, and R. Simino, *Simulation at a maritime container terminal: Models and computational frameworks*, Proceedings of the 22nd European Conference on Modeling and Simulation (Nicosia, Cyprus), June 2008, pp. 261–269.

[43] P. Legato and R.M. Mazza, *Berth planning and resources optimisation at a container terminal via discrete event simulation*, European Journal of Operational Research **133** (2001), 537–547.

[44] P. Legato, R.M. Mazza, and R. Trunfio, *Simulation-based optimization for the quay crane scheduling problem*, Proceedings of the 2008 Winter Simulation Conference (Miami, Florida, USA), 2008, pp. 2717–2725.

[45] P. Legato and M.F. Monaco, *Human resources management at a marine container terminal*, European Journal of Operational Research **156** (2004), no. 3, 769–781.

[46] P. Legato and R. Trunfio, *An open-source discrete event simulation-based optimization package*, Proceedings of 19th European Modelling and Simulation Symposium (Simulation in Industry) (Bergeggi, Italy), October 2007, pp. 125–132.

[47] ———, *A simulation modelling paradigm for the optimal management of logistics in container terminals*, Proceedings of the 21th European Conference on Modelling and Simulation (Prague, Czech Republic), June 2007, pp. 479–488.

[48] P. Legato, R. Trunfio, and F. Mari, *The relevancy of moves definition in the simulation-based optimization of manufacturing systems*, Proceedings of 19th European Modelling and Simulation Symposium (Simulation in Industry) (Bergeggi, Italy), October 2007, pp. 125–132.

[49] A. Lim, B. Rodrigues, and Z. Xu, *A m-parallel crane scheduling problem with a non-crossing constraint*, Naval Research Logistics **54** (2007), no. 2, 115–235.

[50] Marsan M.A., G. Balbo, and G. Conte, *A class of generalised stochastic petri nets for the performance evaluation of multiprocessor systems*, ACM Transactions on Computer Systems **2** (1984), no. 1, 93–122.

[51] F.J. Matejcik and B.L. Nelson, *Two-stage multiple comparisons with the best for computer simulation*, Operations Research **43** (1995), 633–640.

[52] F. Meisel and C. Bierwirth, *Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal*, Operations Research Proceedings 2005, 2006, pp. 105–110.

[53] P.M. Merlin, *A study of the recoverability of computing systems*, Ph.D. thesis, Department of Information and Computer Science, University of California, Irvine, California, USA, 1974.

[54] L. Moccia, J.-F. Cordeau, M. Gaudioso, and G. Laporte, *A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal*, Naval Research Logistics **53** (2006), 45–59.

[55] R.E. Nance, *The conical methodology: A framework for simulation model development*, Proceedings of the Conference on Methodology and Validation (San Diego, CA, USA), The Society for Computer Simulation, 1987, pp. 38–43.

[56] S. Ólafsson, *Iterative ranking-and-selection for large-scale optimization*, Proceedings of the 1999 Winter Simulation Conference (Phoenix, Arizona, USA), December 1999, pp. 479–485.

[57] S. Ólafsson and J. Kim, *Towards a framework for black-box simulation optimization*, Proceedings of the 2001 Winter Simulation Conference (New Orleans, Lousiana, USA), December 2001, pp. 300–306.

[58] ———, *Simulation optimization*, Proceedings of the 2002 Winter Simulation Conference (San Diego, California, USA), December 2002, pp. 79–84.

[59] C.M. Overstreet and R.E. Nance, *Characterizations and relationships of world views*, Proceedings of the 2004 Winter Simulation Conference (Piscataway, New Jersey, USA), 2004, pp. 279–287.

[60] Y.-M. Park and K.H. Kim, *A quay crane scheduling method considering interference of yard cranes in container terminals*, OR Spectrum **25** (2003), 1–23.

[61] F. Parola and A. Sciomachen, *Intermodal container flows in a port system network: analysis of possible growths via simulation models*, International Journal of Production Economics **97** (2005), 75–88.

[62] C.A. Petri, *Kommunikation mit automaten*, Ph.D. thesis, Schriften des Institutes ft Instrumentelle Matematik, Bonn, Germany, 1962.

[63] L. Pi, Y. Pan, and L. Shi, *Hybrid nested partitions and mathematical programming approach and its applications*, IEEE Transactions on Automation Science and Engineering **5** (2008), no. 4, 573–586.

[64] M. Pidd, *Simulation software and model reuse: A polemic*, Proceedings of the 2002 Winter Simulation Conference (San Diego, California, USA), December 2002, pp. 722–775.

[65] ———, *Computer simulation in management science*, 5th ed., John Wiley & Sons, 2004.

[66] ———, *Simulation worldview – so what?*, Proceedings of the 2004 Winter Simulation Conference (Blacksburg, Virginia, USA), December 2004, pp. 288–292.

[67] M. Pidd and R.B. Castro, *Hierarchical modular modelling in discrete simulation*, Proceedings of the 1998 Winter Simulation Conference (Washington, D.C., USA), December 1998, pp. 383–389.

[68] M.L. Pinedo, *Scheduling: Theory, algorithms and systems*, Prentice Hall, 1995.

[69] ———, *Planning and scheduling in manufacturing and services*, Springer Series in Operations Research, Springer, 2006.

[70] A.A. Prudius, *Adaptive random search methods for simulation optimization*, Ph.D. thesis, H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA, 2007.

[71] A.A. Prudius and S. Andradóttir, *Simulation optimization using balanced explorative and exploitative search*, Proceedings of the 2004 Winter Simulation Conference (Piscataway, New Jersey, USA), December 2004, pp. 545–549.

[72] Y. Rinott, *On two-stage selection procedures and related probability inequalities*, Communications in Statistics – Theory and Methods **A7** (1978), 799–811.

[73] P. Rogers, *Optimum-seeking simulation in the design and control of manufacturing systems: experience with optquest for arena*, Proceedings of the 2002 Winter Simulation Conference (San Diego, California, USA), December 2002, pp. 1142–1150.

[74] S. Ronald, *More distance functions for order-based encodings*, Proceedings of the IEEE Conference on Evolutionary Computation, IEEE Press, 1998, p. 558563.

[75] V. Roso, *Emergence and significance of dry ports*, Proceedings of the World Conference on Transport Research (Berkeley, California, USA), June 2007.

[76] M. Sammarra, J.-F. Cordeau, G. Laporte, and M. F. Monaco, *A tabu search heuristic for the quay crane scheduling problem*, Journal of Scheduling **10** (2007), 327–336.

[77] R. Sargent, *Event graph modelling for simulation with an application to flexible manufacturing systems*, Management Science **34** (1988), no. 10, 1231–1251.

[78] ———, *Modelling queuing systems using hierarchical control flow graph models*, Mathematics and Computers in Simulation **44** (1997), no. 3, 233–249.

[79] L.W. Schruben, *Simulation modelling with event graphs*, Communications of the ACM **26** (1983), no. 11, 957–963.

[80] L.W. Schruben and T.M. Roeder, *Fast simulations of large-scale highly congested systems*, Simulation **79** (2003), no. 3, 115–125.

[81] A.F. Seila, *Spreadsheet simulation*, Proceedings of the 2006 Winter Simulation Conference (Monterey, California, USA), December 2006, pp. 11–18.

[82] L. Shi, C.-H. Chen, and E. Yücesan, *Simultaneous simulation experiments and nested partition for discrete resource allocation in supply chain management*, Proceedings of the 1999 Winter Simulation Conference (Phoenix, Arizona, USA), December 1999, pp. 395–401.

[83] L. Shi and S. Ólafsson, *Nested partitions method for stochastic optimization*, Methodology and Computing in Applied Probability **2** (2000), no. 3, 271–291.

[84] ———, *Stopping rules for the stochastic nested partitions method*, Methodology and Computing in Applied Probability **2** (2000), no. 1, 37–58.

[85] K. Sörensen, *Distance measures based on the edit distance for permutation-type representations*, Journal of Heuristics **13** (2007), no. 1, 35–47.

[86] R. Stahlbock and S. Voß, *Operations research at container terminals: a literature update*, OR Spectrum **30** (2008), 1–52.

[87] D. Steenken, S. Voß, and R. Stahlbock, *Container terminal operation and operations research - a classification and literature review*, OR Spectrum **26** (2004), 3–49.

[88] J.R. Swisher, P.D. Hyden, S.H. Jacobson, and L.W. Schruben, *A survey of simulation optimization techniques and procedures*, Proceedings of the 2000 Winter Simulation Conference (Orlando, Florida, USA), December 2000, pp. 119–128.

[89] UNCTAD, *Review of maritime transport*, Tech. report, United Nations, New York and Geneva, 2004.

[90] ———, *Review of maritime transport*, Tech. report, United Nations, New York and Geneva, 2007.

[91] I.F.A. Vis and R. De Koster, *Transshipment of containers at a container terminal: an overview*, European Journal of Operational Research **147** (2003), no. 1, 1–16.

[92] R.R. Wilcox, *A table for rinott's selection procedure*, Journal of Quality Technology **16** (1984), no. 2, 97–100.

[93] E. Yücesan and L.W. Schruben, *Structural and behavioral equivalence of simulation models*, ACM Transactions on Modelling and Computer Simulation **2** (1992), no. 1, 82–103.

[94] W.Y. Yun and Y.S. Choi, *A simulation model for container-terminal operation analysis using an object-oriented approach*, International Journal of Production Economics **59** (1999), 221–230.

[95] B.P. Zeigler, *Theory of modeling and simulation*, John Wiley and Sons, 1976.

[96] ———, *Multifacetted modelling and discrete event simulation*, Academic Press (1984).

[97] ———, *Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems*, Academic Press (1990).