A Simulation-Based Optimization Framework for Parameter Optimization of Supply-Chain Networks

ARTICLE *in* INDUSTRIAL & ENGINEERING CHEMISTRY RESEARCH · MARCH 2006

Impact Factor: 2.59 · DOI: 10.1021/ie051121g

4 AUTHORS, INCLUDING:

Fernando Mele
National University of Tucuman
44 PUBLICATIONS **544** CITATIONS
SEE PROFILE

Antonio Espuña
Polytechnic University of Catalonia
217 PUBLICATIONS **1,850** CITATIONS
SEE PROFILE

Luis Puigjaner
Polytechnic University of Catalonia
450 PUBLICATIONS **5,089** CITATIONS
SEE PROFILE

A Simulation-Based Optimization Framework for Parameter Optimization of Supply-Chain Networks

Fernando D. Mele, Gonzalo Guillén, Antonio Espuña, and Luis Puigjaner*

Chemical Engineering Department, Universitat Politècnica de Catalunya, ETSEIB, Av. Diagonal 647, E-08028 Barcelona, Spain

This work presents a novel approach that addresses the management of chemical supply chains (SCs) under demand uncertainty. One of the main objectives is to overcome the numerical difficulties associated with solving the underlying large-scale mixed integer nonlinear problem (MINLP). The approach that is proposed relies on a simulation-based optimization strategy that uses a discrete-event system to model the SC. Within this framework, each SC entity is represented as an agent whose activity is described by a collection of states and transitions. The overall system is coupled with an optimization algorithm that is designed to improve its operation. This strategy is a very attractive alternative in the field of decision-making processes under uncertainty, the advantages of which are highlighted with some cases of SC networks that are composed of several plants, warehouses, distribution centers, and retailers.

1. Introduction

The chemical process industry (CPI) operates in a rapidly changing and uncertain environment. Uncertainties appear in customer demands for products, in the prices of raw materials and products, in lead times for the supply of raw materials and the production and distribution of the end products, and in process and quality failures, for example.[1]

Generally, supply-chain management (SCM) involves decision-making related to the administration of resources throughout the entire supply chain (SC), from the initial suppliers to the end consumers, and including the manufacturing, storage, and distribution centers.[2] Specifically, operating in a CPI environment involves SCM decisions at different levels, which are made to satisfy multiple goals. For instance, depending on their budget planning procedures, companies may decide to expand or contract their production capacity once a year. Several times during this long-term period (e.g., once a month), they must also decide the tuning to be used in inventory control policies, and, more frequently (e.g., once a week), they may have to schedule production to satisfy customer demand. This type of problem, which is referred to as sequential decision-making under uncertainty, arises in many decision-making contexts and in a variety of situations in which one makes decisions at different stages in response to uncertainty that is resolved over time. This class of problem has attracted increasing attention from the research community. Moreover, various theoretical and computational works have been devoted to modeling and solving it. The resulting complexity, from theoretical and computational perspectives, is a challenge for both practitioners and researchers.

Current approaches to this problem transform it into several optimization problems whose structures are extremely complex (nonlinear, nonconvex problems with mixed integer and continuous variables). Specifically, considering the uncertainty associated with the SC and modeling heuristic rules that drive the operation of the nodes embedded in the network results in large-scale mixed-integer nonlinear programming (MINLP) problems. Solving these problems using mathematical programming techniques requires excessive computing time, because of their size and complexity when real scenarios are considered. In this work, a novel approach is proposed to address the sequential decision-making problem under uncertainty in the area of chemical SCs, which has the objective of overcoming the computational limitations of current solution methods. This approach considers two important components: an agent-based model used to represent the SC network and a metaheuristic optimization algorithm that is designed to improve the operation of this network.

The paper is organized as follows. First, the literature review on the three key points involved—agent-based modeling, decision-making under uncertainty, and simulation-based optimization—is given. Next, a motivating case study is used to outline the problem and present suitable approaches for solving it. The method proposed is then explained in detail. Finally, the results are discussed and general guidelines are provided for future work.

2. Literature Review

Taking into account the concepts and tools presented in this paper, the review is divided into three interrelated sections: agent-based modeling, decision-making processes under uncertainty, and simulation-based optimization.

2.1. Agent-Based Modeling. Software agent-based systems have been determined to be effective tools for solving certain problems of a complex nature. Because they are built by combining autonomous computer programs or agents, they typically perform significantly better than the individual programs do in isolation. The main reason for this improvement lies in the cooperation and distribution of the tasks between the individual agents that constitute the system.

The agent-based models for SCM can be considered to be models for dynamic simulation, i.e., executable *descriptive models* that are used to better understand functional relationships within a system and between the system and the environment, by means of experiments. Research in the area of dynamic simulation dates back to the 1960s, and since then, it has evolved into the object-oriented paradigm and agent-oriented technology that we know today.

* To whom all correspondence should be addressed. Telephone: +34-93-401 66 78. Fax: +34-93-401 09 79. E-mail: Luis.Puigjaner@upc.edu.

10.1021/ie051121g CCC: $33.50
Published on Web 03/16/2006

2.1.1. Agents Applications in the Process Industry. The need for coordinating multiple activities or tasks is a key feature of many chemical engineering problems, to which multi-agent systems can offer well-suited answers. Process design is an interesting example of an agent-based application. Han, Douglas, and Stephanopoulos[3] decomposed the entire design process into tasks and assigned an agent to perform each one. Maguire et al.[4] also proposed an agent-based environment for process design. This modeling environment was applied by the authors[5] to model a flash process. Siirola, Hauan, and Westerberg[6,7] proposed a modular framework to implement agent-based systems for engineering design. Through a variety of different algorithmic agents, the key ideas are shown, using a multi-objective optimization problem as an example.

The work by Eo et al.[8] is a good example of applications to plant operations. Agent technology has also been applied in simulation tools to solve operations research problems, such as operations scheduling.[9,10]

Agent systems' other skills, such as negotiation and physical distribution, have also been examined. McGreavy et al.[11] proposed an agent architecture for concurrent engineering. They defined three main agents (process, control, and equipment design) that exchange design data and negotiate for improvement in the complete design process. Batres and co-workers[12,13] also addressed a concurrent process engineering problem in which agents that perform different design tasks were distributed over a local network that resided in different computers.

Because of the growing interests in agents, many specifications and protocols have been defined for agent communication, interaction, and internal activity, with the intention of making these into standards. Such is the case for the Foundation for Intelligent Physical Agents (FIPA, www.fipa.org), which has been working in this field since 1996, and the European Coordination Action for Agent-Based Computing (www.agentlink.org), which periodically publishes works on agent developments, including SCM.

2.1.2. Agents Applications in SCM. In recognition of the geographically distributed nature of a SC network and the requirements of intelligent decisionmaking, the area of simulation of SC networks has been given increasing amounts of attention by researchers and practitioners.[14−16] Compared to the many efforts that are made in other types of approaches to SCM, the work on agent-based systems is still very limited, especially that devoted to the CPI. The current situation is such that most approaches focus on partial problems. Furthermore, the major drawback of simulation models is a lack of optimization skills, and thus external optimization is often needed (see Section 2.3).

The earliest attempts to use dynamic simulation in production chains was reported by Forrester,[17] who was driven to perform a dynamic simulation of industrial systems by means of discrete time mass balances and nonlinear delays. However, because of the complexity of the models and the computer limitations at that moment, the work only covers small, academic examples.

In the past few years, multi-agent systems have become a promising tool for solving SC problems, while being a natural result of the progress in the field of dynamic simulation. Goodwin et al.[18] presented a framework for providing decision support in an on-line exchange. A multi-agent system finds matches of demand and supply and provides the user with the best set of transactions. It is one of the many attempts to model the exchanges on the Internet by mimicking the negotiation processes.

Other researchers have studied agents by focusing on cooperation between SC systems. Sauter, Parunak, and Goic[19] presented an architecture called Agent Network for Task Scheduling (ANTS). In this architecture, agents represent the different elements in the SC. Fox, Barbuceanu, and Teigen[20] (www.eil.utoronto.ca/iscm-descr.html) simulated the three levels of decisionmaking: strategic (production allocation and sourcing strategy), tactical (forecasting and planning), and operational (inventory deployment and detailed scheduling). Swaminathan, Smith, and Zadeh[21] (www-2.cs.cmu.edu/afs/cs/project/ozone/www/supply chain/supply chain.html) presented a modeling and simulation framework for developing customized decision support tools for SC re-engineering. Agents represent SC entities, e.g., customers, manufacturers, and transportation. These agents use different interaction protocols to simulate the SC flows. They also use various control policies to manage inventories, procure components, and determine optimal transportation routes. A similar, more recent contribution in this direction is the work performed by Julka, Srinivasan, and Karimi.[22] They developed a framework that had two basic elements: object modeling of SC flows and agent modeling of SC entities. The user can configure SC scenarios and compare the effect of different decisions using various performance measures. However, apart from some preliminary works such as that by Mele et al.,[23] these approaches do not include resources for global improvement or optimization.

In addition, a variety of commercial software packages for SCM indicate the growing importance of this area,[24,25] although almost none of them are based on software agents. Most commercial tools are enterprise resource planning (ERP) systems whose capabilities have been extended, because the emphasis is on automating transaction processing.

2.2. Decision Processes under Uncertainty. In regard to decision-making processes under uncertainty, many advances have been observed in the supporting theory, including algorithmic developments and computational capabilities for solving this class of problems, most of which fall into one of two approaches: multistage stochastic programming and stochastic optimal control.

Multistage stochastic programming considers the problem that involves a sequence of decisions reacting to events that occur over time. At each stage, decisions are based on past observations and decisions before the future events occur.[26] To avert the problems encountered when one is working with continuous distributions, a finite set of scenarios is often generated to represent the space. However, when the reduced space is specified, the stochastic program becomes a deterministic equivalent program, the size of which can easily grow out of hand for a large number of scenarios, making the direct solution approaches numerically intractable, thus requiring methods of decomposition or aggregation.[27]

Stochastic optimal control describes a sequential decision problem in which the decisionmaker chooses an action in the state involved at any decision stage according to a decision rule or policy. Dynamic programming provides a framework for studying such problems and designing algorithms to compute an optimal control policy. Cheng, Subrahmanian, and Westerberg[28,29] developed a multi-objective stochastic dynamic programming algorithm based on the ϵ-constraint method to solve multi-objective Markov decision problems. However, for large problems, dynamic programming also suffers numerically from dimensionality. Therefore, several approximate approaches have been proposed to improve the results of solving large-scale problems, such as a simulation-based optimization framework in which the optimizer uses a multi-objective evolutionary

algorithm to search simultaneously for a diverse set of parameters that defines Pareto optimal policies in a single run.

These authors[30] claimed that both approaches—stochastic programming and optimal control—are able to solve the same class of problems with different purposes and emphasis. Both are essentially equivalent, because they can, in principle, determine the same solution to the same problem. However, they exhibit differences in formulation and solution, with the consequent advantages and disadvantages for specific problems.

Efficient numerical solution proposals can be achieved by combining several techniques that belong to each approach. However, the resulting strategy cannot simply be adapted to solve general problems, and problem-specific approximations or heuristic-based methods are usually required. The works by Cheng, Subrahmanian, and Westerberg[30] and Jung et al.[31] are relevant examples of these combinations.

2.3. Simulation-Based Optimization. Simulation-based optimization is an attractive combined strategy, similar to those previously mentioned. It involves the situation in which the analyst wishes to find which of many possible sets of input parameters lead to the optimal performance of a system. Thus, a simulation model can be understood as a function whose explicit form is unknown and which converts input parameters to performance measures.[32] Simulation-based optimization is an active area of research in the field of stochastic optimization. Reviews of the current research in this area can be found in Fu.[33] In the literature on process system engineering, simulation-based optimization approaches for SCM have received little attention and are currently awaiting further study. However, works by Subramanian, Pekny, and Reklaitis[1] and Jung et al.[31] are well-regarded contributions to this field.

There are a variety of ways of optimizing processes represented by means of a simulation model. Many of them have not been used to develop optimization for simulation software, because they often require a considerable amount of technical sophistication on the part of the user, as well as a substantial amount of computation time.[34] This is closely related to the fact that some of these techniques are local search strategies that may be highly problem-dependent. The success of metaheuristics in this field is perhaps that they are designed to seek global optimality and their properties are apparently robust in practice, even though they do not yet have a sound theoretical basis. A metaheuristic can be defined as a high-level algorithmic framework that guides other heuristics in a search for feasible solutions.

Evolutionary algorithms are a particularly important subset of metaheuristic methods, whose main advantage over other metaheuristics is that they are capable of exploring a larger area of the solution space with a smaller number of objective function evaluations. Because, in the context of simulation-based optimization, evaluating the objective function entails running the simulation model, being able to find high-quality solutions early on in the search is of critical importance. Evolutionary algorithms, among them genetic algorithms (GAs), have been used to optimize multimodal, discontinuous, and nondifferentiable functions. In the field of SCM, the approaches are supported by the research in GAs combined with mathematical programming.[35−37] However, these works mostly involve strategic decisions (for instance, combinatorial operation research problems such as the multistage facility location/allocation problem, in which the decisions are related to the facilities to be opened or the distribution network to be used), rather than tactical or operational decisions. Moreover, even though rapid progress is being made in this area, the problem is usually different from the one undertaken here.

3. Motivating Problem

3.1. Problem Description. In this section, a general description of the problem, based on the research of Cheng, Subrahmanian, and Westerberg,[30] and the candidate solution strategies to be used are presented. The problem is enunciated as a discrete-time decision process over a given time horizon, in which managers periodically make decisions based on the available information. A partition of the set of decisions is performed according to the frequency with which they are updated. The system evolves over time and generates values for some performance indexes. The problem then becomes one of finding the decision strategies that can optimize these performance indexes.

The SC of a company is now considered. It uses different resources to produce products j in a time horizon of T periods. The company has the option to change the parameters that feature the inventory control laws at the beginning of each period $\theta \in \{1, ..., T\}$. Each period θ is divided in L subperiods. Inventory levels are reviewed and production planning decisions are made at the beginning of each subperiod $\tau \in \{1, ..., L\}$, given that the size of τ is less than the size of θ.

The uncertain parameters, such as product demand, are modeled as a discrete time stochastic process that gives a value of the uncertain variable at some instants in the time horizon. Hereafter, one realization of the uncertainty is identified with ω.

Decisions are made in stages as information arrives and uncertainty is unveiled over time. The following sequence summarizes the events occurrence during each period, as illustrated in Figure 1:

(*a*) At the beginning of each period $\theta = 1, ..., T$, having observed the information available, new values for the inventory law parameters η are chosen.

(*b*) At the beginning of each subperiod τ, and based on the current information, e.g., inventory level, the company plans production at the manufacturing plants and prepares deliveries of materials in the case of warehouses and distribution centers, to replenish the inventory up to a certain level. Demand in subperiod τ may arrive and is satisfied as much as possible. The exceeding inventory is carried over to the next subperiod and unsatisfied demand is backlogged as cumulated orders.

(*c*) The procedure is repeated from step *a* (rolling horizon) until $\theta = T + 1$.

It is important to notice that this problem is embedded within a wider decision-making problem. There might be long-lasting decisions, strategic decisions, to update at the beginning of each period $h \in \{1, ..., H\}$, with the size of h being greater than the size of θ (for instance, plants or warehouses capacities), such that the scope of these decisions would include the decisions previously described. Moreover, the decision of tuning inventory control laws is just an example of the types of decisions that can be considered.

The sequence of decisions made will represent a decision strategy. Decisions are made at each stage on the basis of past observations and decisions. In other words, a decision strategy is non-anticipatory, in the sense that it cannot be dependent on the specific future events, which are not yet realized.

At each stage of the decision process, although the state variable values change as a consequence of the selected decision, the company receives a profit or incurs on a cost. After choosing and implementing the decision strategy, the decisionmaker

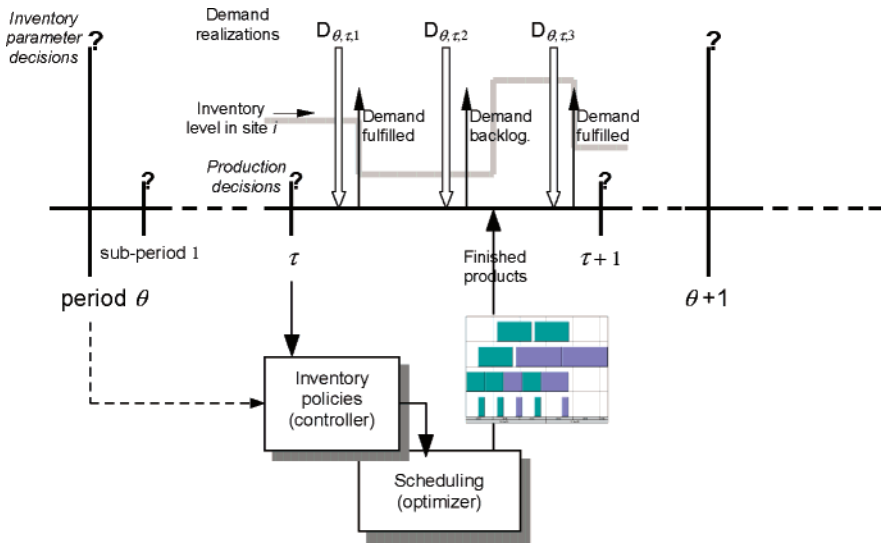

Figure 1. Decision-making process under uncertainty problem.

receives the outcomes for the system performance. The manager's motivation is to choose a decision strategy such that these outcomes are as good as possible, that is to say, optimal. This necessitates the definition for performance measures, or optimality criteria, to compare alternative decisions. In this work, the optimality criteria considered is the expected total profit. The expected total profit under a given decision strategy η can be expressed as follows:

$$\mathcal{F}(\eta) = E[\text{Profit}(\eta,\omega)] \tag{1}$$

where the expectation $E[\cdot]$ is taken with respect to all random variables in ω.

3.2. Solution Methodologies. The problem structure described in the previous section permits the two major solution strategies commented previously: multistage stochastic programming and stochastic optimal control.

In regard to multistage stochastic programming, these models give adjusted decisions at each time stage, based on the uncertainty unveiled so far. The rigorous solution of this multistage stochastic program, using mathematical programming tools, could be undertaken. However, routine solution of this type of problem, given the unavailability of methods to properly examine continuous probability distributions, is currently impractical. To avoid this problem, a finite set of scenarios can be generated by sampling or using a discrete approximation of the given distributions. In this case, the evolution of random parameters over time can be represented by a treelike structure. Each path through the tree from its root (origin node) to one of its branches corresponds to one scenario or sample path, which represents the sequence of realizations of random parameters. The construction of a deterministic equivalent using the scenario approximation will create a problem much beyond the power of current nonlinear (NLP) or MINLP solution techniques, to say nothing of the additional dimensionality increase when the detailed scheduling variables and constraints are added to the formulation.

Regarding the stochastic optimal control approach, it is possible to consider the problem as a Markov decision process,[30] in which decisionmakers periodically review the state of the system and make decisions according to control policies that prescribe the action for each possible state. Within this research stream, the numerical intractability of the rigorous algorithm for large problems needs approximation approaches. The intractability, in this case, is related to the size of the state space, especially when the states are continuous and multidimensional. An approximate alternative to overcome the numerical difficulties associated with the aforementioned techniques is a simulation-based optimization framework to find the parameters values that optimize an objective function using a rolling horizon control scheme.

Having this in mind, an approximation framework is proposed in this work that exploits the advantages of both approaches to find the optimal first stage(s) decisions or policies. This framework is described in detail in the next section.

4. Proposed Framework

SCs, similar to many other real-world systems, are too complex to allow realistic models to be evaluated analytically; thus, an approach that considers the SC by means of agents and discrete event-based dynamic simulation[38] is very well-suited to analyze these systems. Compared to analytical techniques, simulation provides the flexibility to accommodate arbitrary stochastic elements and generally admits modeling the complexities and dynamics of real world SCs without having to perform excessive simplifying assumptions. Although system managers were able to play "what if" scenarios with input data and simulation models to evaluate alternative solutions, an optimization algorithm may help to eliminate the need for random trial and error and allows the automation of the optimization procedure. Thus, even if multi-agent systems are dynamic simulation models, they are also able to integrate optimization/improving policies at different levels. Hence, they are not only descriptive tools but they also are related to operations research and control-based approaches.

In this work, an agent-based simulation model accurately representing the SC operation is coupled with a GA to optimize certain parameters in regard to the operation of the agents. The logical (or heuristic) rules, which are implemented in the agents to respond to the arriving events, are parametrized, and the metaheuristic algorithm is then invoked to act over these parameters, in search of an improvement in the objective function. The motivation for using GAs is to take advantage of the features of this technique, applying it to complex problems such as that previously described (see Section 3.1). Particularly, in the presented examples, the framework permits one to obtain the parameters values associated with the inventory control policies applied in the involved entities, in such a way that the

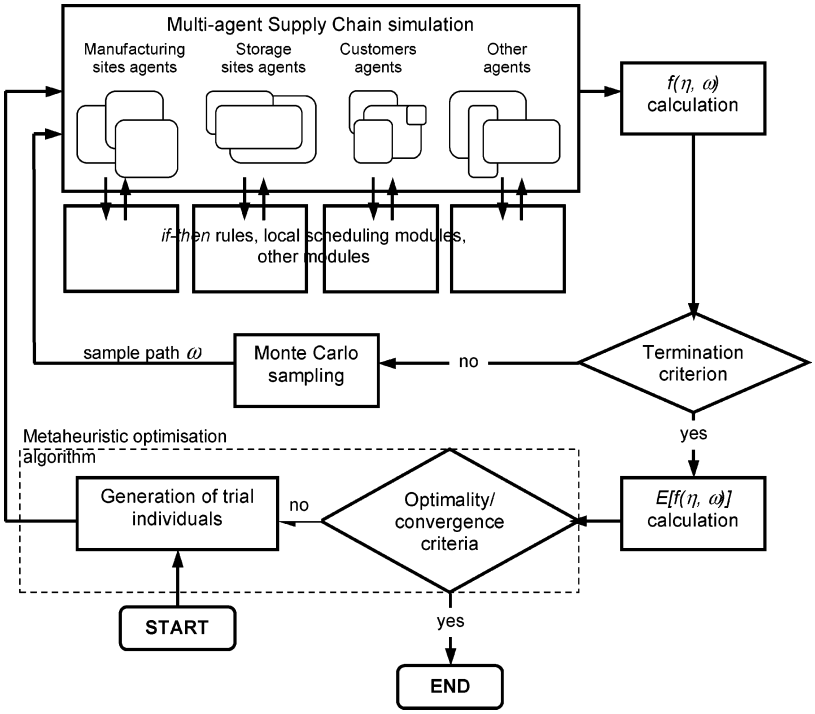

Figure 2. Proposed framework.

SC profit is optimized over a given time horizon. Nevertheless, the proposed approach is general enough to enable the choice of any other parameter for optimization (equipment capacity, transport policy, inventory policy selection, backlogged orders management, etc.), as well as any other objective function.

The contribution of this work, compared to the previous work, is not only in the hybrid strategy proposed (simulation/optimization) but especially in the use of an agent-based simulation model. As described previously, this is a highly suitable and flexible means of representing the SC behavior.

The proposed approach applied to the problem presented previously is summarized in Figure 2. First, and to determine the value of the performance index, $f(\eta,\omega)$, demand scenarios extending over the entire planning horizon ω are generated in a Monte Carlo manner. A simulation run of the SC is then executed, implementing a specific inventory control system for all the nodes in the SC network, for the full horizon, and for a given scenario. The result of such multiple simulations serves to provide an estimate of the objective function, $E[f(\eta,\omega)]$. The parameters for tuning the inventory control policies are then adjusted by means of an outer optimization loop. The outer optimizer uses the parameters of the inventory control laws as decision variables, whereas the inner problem involves a simulation with a series of embedded scheduling subproblems.

Next, the agent-based SC simulator and the two key subproblems, the outer optimization subproblem and the inner planning subproblem, are defined, discussing in more detail the various computational details that are needed to link these subproblems and drive the computations.

4.1. The Agent-Based SC Simulator. A software agent-based system has been developed to implement a SC model. This model specifies, in a given way, the behavior that individual participants are expected to have in the system and avoids having to solve a set of differential equations. In this paper, it is proposed to use a model that includes all the entities that belong to the SC as independent and well-defined agents, each of these being represented by a collection of states and transitions as an event-driven system.

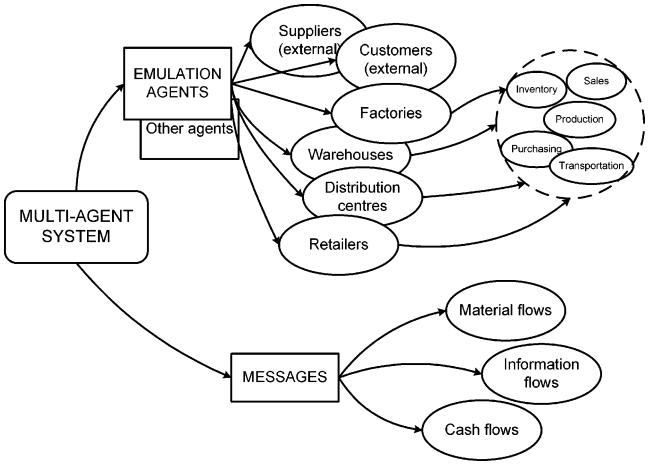

Figure 3. Multi-agent system.

Because there is no general consensus about the definition of software agents, in this work, the definition adopted is the widely accepted one from Wooldridge:[39] *An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.*

The agent-based simulator consists of two categories of objects: the software agents and the messages. All agents communicate with each other by sending objects of *message* class. The behavior of an agent is described in the form of one or more *tasks* classes, which are coded in the agent. An agent may perform multiple tasks with multiple threads active in a task. Among the agents that participate in this system (see Figure 3), the *emulation* agents represent the physical entities, facilities or nodes, in the real-world SC such as suppliers, customers, manufacturing plants, distribution centers, etc., and they are described next. Each of these agents includes several subagents. Moreover, the system has other types of agents apart from emulation agents. Among them, the most important is the central agent, which has no correspondence with a real entity in the chain. With respect to the messages, they are objects that are

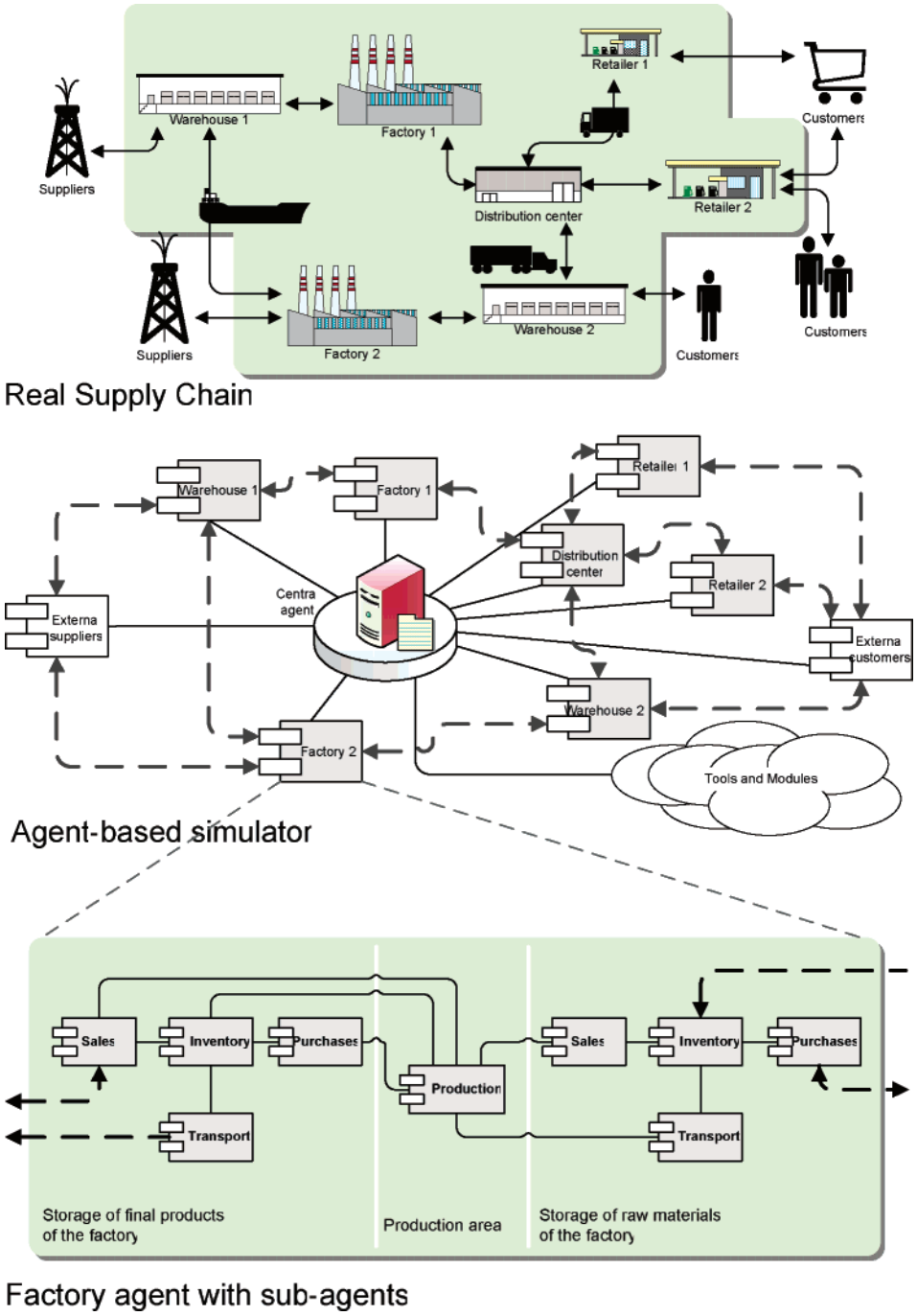

Figure 4. Agent modeling of the supply chain (SC) entities.

exchanged between the agents and they represent material, information, or cash flows.

Figure 4 shows how the real system is modeled using agents (one for each site, plus a central coordination agent), with each of them being modeled, in turn, by a set of subagents representing the internal departments of these entities. In this figure, the internal structure of an agent factory is shown.

4.1.1. Software Agents. The emulation agents can represent external entities, customers or suppliers, or any entity belonging to the SC network. The agents that represent external customers or suppliers emulate the behavior of real or hypothetical entities to consider them in the simulation runs. Therefore, these agents have specific models mimicking this type of entity. Otherwise, the other emulation agents are called *site* agents. This is an important point from the implementation perspective, because it is possible to define a generic agent or class for every SC site and then generate from this class some children classes, by inheritance, with the specific features for representing factories, distribution centers, or whatever entity of interest. A *state and transitions* diagram may be used to represent the agent tasks. These diagrams are made of nodes and links. Nodes represent states of a system, each one with different steps that may contain small programs. A link, arc, or transition marks the change of the system from one state to another. It represents an event or condition, which is necessary so that the system switches its state.

The following list describes each type of emulation agent.

4.1.1.1. External Customer Agent. This agent represents outside customers that issue orders to the SC being modeled. Orders are generated on the basis of customer demand, which may be modeled as a sequence of specific customer orders (obtained from historical records), as an aggregated demand over

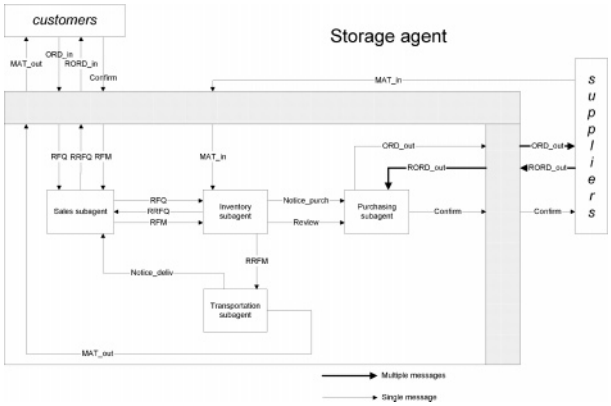

Figure 5. Generic site agent with subagents and messages.

a period of time using forecasting techniques or as data sampled from a distribution of probability.

4.1.1.2. Site Agent. A site is defined as a particular location in the SC network that can deliver a set of products from a set of input materials by utilizing a set of technologies. This definition is general enough to include a manufacturing plant or simply a warehouse or packaging site. Site agents have subagents that perform internal tasks at a site. Figure 5 shows a site agent with the corresponding subagents and messages between them. Upon receiving messages from other sites, this agent decides the further course of action. Let us show, in some detail, one of the possible processes. For example, if a request for quote (RFQ) is received (message "ORD_in" in Figure 5), the agent forwards the message to its sales subagent (message "RFQ"). It then creates an internal request for information and sends it to the inventory subagent (the RFQ message between the sales and inventory subagents). The inventory responds to the sales subagent through a "RRFQ" message, which is broadcast to the customer. Depending on the content of this message and the customer requirements, the customer could send a "Confirm" message, which will arrive to the sales subagent as a "RFM" message, which is then sent to the inventory. The inventory depletes its level in the required quantity and sends a message to the transportation subagent, which delivers the indicated quantity to the customer through a "Mat_out" message. Transportation also informs the sales subagent about the delivery by posting the "Notice_deliver" message.

The site agents that belong to the proposed multi-agent system have five subagents: sales, inventory, production, purchasing, and transportation.

The *sales subagent* models the sales department of the site. It is responsible for processing of RFQs and subsequent orders received by the site. It keeps track of external demand and prepares demand plans. It also keeps a record of all of the orders that are received, fulfilled, or queued. In the case of made-to-stock goods, the sales subagent is the entity that generates the order.

The *inventory subagent* models the materials-handling department of the site. It issues the required commodities to other subagents and replenishes the raw materials stock through procurement from other locations. This framework enables the incorporation of different inventory policies whose modeling involves many parameters such as safety stocks, reorder points, material handling resources, storage types, and capacities, which has been used in this work to demonstrate the chance for improving and optimizing the SC performance.

The *production subagent* models the operations or production department of a location. It can have additional subagents that model the various production lines or facilities of a plant. This subagent awards received orders to the production lines. In this work, upon receiving a RFQ, it calls for a scheduling program implemented in GAMS,[40] which gives the arrival time of the batch of each product to the inventory of finished products, and it generates orders for updating the inventory of raw materials. Thus, the production subagent also procures materials from the warehouse and allocates them to the lines based on their needs.

The *purchasing subagent* manages the purchasing tasks of the site agents. It decides when to ask for material, by sending requests to all the possible suppliers, and then choosing an option among the received offers. Thus, the purchasing subagent may also act by attaching the site to a network of suppliers. It involves long-term and short-term negotiation decisions. Figure 6 illustrates how the purchasing subagent of a storage facility choses between the offers for a given product provided by several factories (e.g., a piece of C# code and a flowchart). The purchasing subagent asks for product as it receives a message for inventory replenishment from the inventory subagent. The purchasing subagent then asks the central agent for the list of offers made by the potential suppliers. Upon receiving this list,

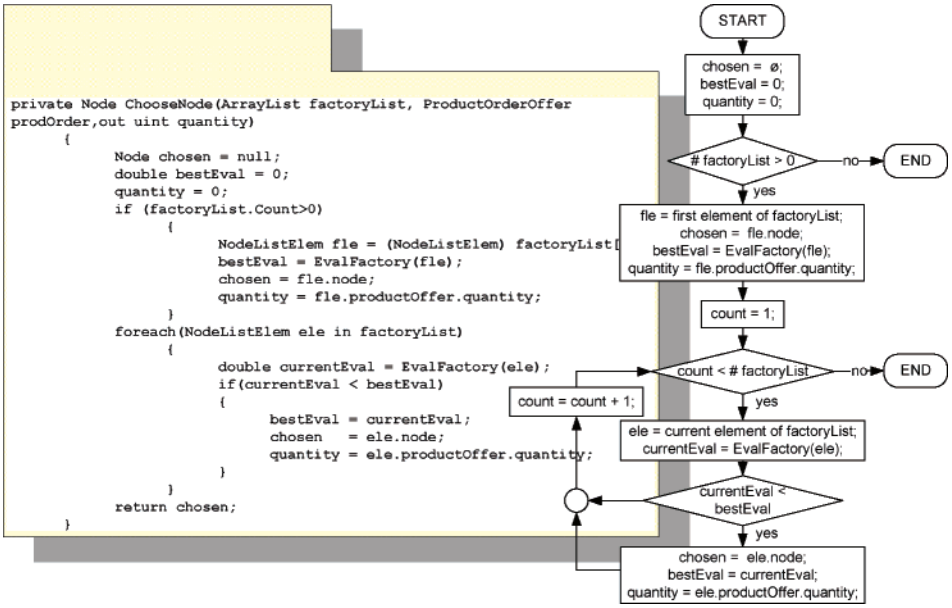

Figure 6. Sample of the code for the activity of the purchasing subagent.

it chooses the best offer as follows. If the number of items in the list is zero, the algorithm finishes. Otherwise, the first element in this list is chosen as the best supplier. The system then enters a loop in which this supplier is compared with all the other options in the list, in terms of availability (in quantity and time) and cost, to choose the best supplier of the list. This chosen option will be the supplier that will provide the necessary materials.

The *transportation subagent* models transportation time, vehicle loading, and transportation costs. Order batching policies (by weight or volume), materials handling resources, and transportation resources owned by this site may be specified. The model of this agent can be considerably simplified if a contract with a third-party logistic is assumed.

4.1.1.3. External Supplier Agent. This agent represents outside suppliers that send materials to the SC being modeled. Usually, as observed with the external customer agent, this one has an approximated model of the external suppliers of the chain. This model is necessary to (i) represent the supplier behavior and then (ii) be able to study the negotiation processes with suppliers.

The central agent is a nonemulating agent that acts as a coordinator of the system. Its mission is related to the communication among the emulating agents as well as the coordination with the analysis and optimization tools.

4.1.2. Modules. The agent-based system uses several modular tools to perform specific tasks or to get the values of specific indexes. There are six different modules.

(1) A forecasting module, which provides accurate values for simulating the customer demand.

(2) A negotiation module, the objective of which is the signing of long-lasting contracts with external suppliers and customers.

(3) Three modules that provide the value of SC performance indicators, namely economical indicators (profit, costs), financial indicators, and environmental impact indicators.

(4) An optimization module based on metaheuristic techniques that takes into consideration the values of the performance indicators to improve the SC operation.

According to the scope of this paper, only the profit calculation and optimization modules will be considered.

4.2. Outer Optimization Subproblem. The objective function of the outer optimization involves a stochastic optimization in which the profit is maximized by choosing the variable (inventory) parameters η.

$$\max_{\eta} \mathcal{F}(\eta) \tag{2}$$

Equation 2 is evaluated in the inner loop using the results of multiple Monte Carlo samplings with embedded discrete event simulation, which, in turn, involves inventory control, if−then rules, and scheduling optimizations.

The outer optimization algorithm consists of a GA-based strategy, to improve the values of the variable parameters. By utilizing an agent-based simulator to represent the SC, preliminary experiments have shown the nonconvex nature of the objective function in the solution space on the model input variables, not only for the inventory parameters but for any other input variables. In addition, the unavailability of explicit equations to model the SC behavior has encouraged the GA-based strategy as a suitable tool for performing the stochastic optimization in the outer loop. The scheme is as follows:

Step I. Create an initial population of N randomly generated individuals (a set of inventory parameters), η^k_{gen}, gen = 0, for all products j and facilities i. The term "gen" is the counter for the GA generations, i.e., the outer loop iterations, and the parameter k accounts for each individual in the population.

Step II. Run a sufficient number (n) of Monte Carlo samplings to generate different sample paths ω.

Step III. Execute the simulator with its embedded optimizations and rules to obtain a reliable estimate of the expected profit for each individual k, $\mathcal{F}(\eta^k_{\text{gen}})$.

Step IV. Apply genetic operators over the current population of η^k_{gen} values to generate a new one.

Step V. If gen ≥ MaxGen, with MaxGen being the maximum number of generations, stop. Otherwise, set gen = gen + 1 and go to step II. Until this moment, the best solution found is η^*_{gen}:$\mathcal{F}(\eta^*_{\text{gen}}) = \max\{\mathcal{F}(\eta^k_{\text{gen}}),\ k = 1, ..., N\}$.

4.3. Inner Subproblem. The computations inside the outer loop optimization involve the repeated simulation of the SC operation over the planning horizon, each with a given Monte Carlo sample of the uncertain parameters (in this case, the demand). Within each such simulation, a series of decisions are periodically made to replenish inventory in the different nodes and satisfy customer demand. The simulator has the outstanding advantage of being capable of representing the real-world SC operation as fitted as desirable. For instance, the simulator is able to include the replenishment control policy used by the decisionmakers at each facility as well as the optimization system used to schedule the production in manufacturing plants (see Section 6.1.2). Each complete simulation run travels across a so-called timeline. The procedure for executing a timeline is as follows:

Step I. Run the discrete event simulation for the planning period θ with demand outcomes for the planning period, ω. Demand arrives with a given frequency unleashing a set of events, mostly through if−then rules, that drive the simulation imitating the firm operation in the real world.

Step II. At the beginning of each subperiod τ that belongs to the planning horizon, the simulator emulates the decision-making process: production scheduling (optimization procedure) at the manufacturing plants and inventory replenishment (control procedure) in all of the facilities of the SC network.

Step III. Calculate and record objective function value, Profit(η,ω).

By repeating the aforementioned procedure for a number n of Monte Carlo samples, the performance results necessary to compute the expected value, $\mathcal{F}(\eta) = E[\text{Profit}(\eta,\omega)]$, can be collected. Note that the accuracy of the expected value will be governed by the number of analyzed timelines and their representativeness.

Because the customer demand is an external source of uncertainty, there are two ways to sample the necessary number of scenarios: at the beginning of the calculations, and repeating the sampling process every time the proposed algorithm enters the inner loop. In this work, the second option has been preferred to improve representativeness.

In some cases, the use of the inventory parameters can be considered as a simplification of the original problem from the stochastic multistage programming point of view, because they exclude feasible solutions without assessing their optimality. In other cases, however, the heuristics included are an accurate representation of the real world, because only the solutions that they represent can be practically implemented in a real industrial scenario. In both situations, solutions given by the simulation-based approach generally cannot guarantee optimality. Nevertheless, the proposed approach is capable of providing reasonable solutions, in time and effort, using building blocks that are readily implementable in practice.

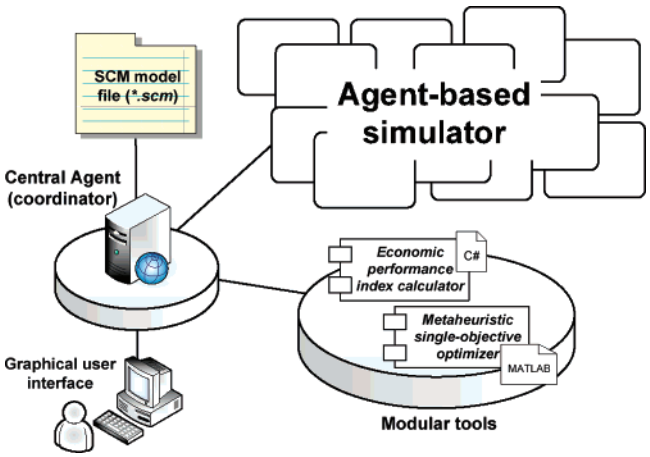

Figure 7. Implementation of the proposed methodology.

5. Implementation of the Proposed Strategy

The computational schema for implementing the proposed strategy uses the framework .NET of Microsoft, and consists of the following elements: the central agent, the agent-based simulator with inventory control systems and scheduling capacities, and the objective function calculation module (see Figure 7).

The central agent provides centralized computation management and tracking of the sequence of actions to conduct the entire strategy. It is implemented using C# as the programming language and its various functional roles include (i) issuing commands for retrieving data to run the strategy, (ii) generating commands for executing the simulation model, (iii) generating demand scenarios, and (iv) saving the calculation results. This module loads the initial necessary data to define a SC case, which are contained in an XML file with an .scm extension. This file includes the following:

(1) The network structure (the number of entities, products, and processing units in the chain, their locations and the relationships between them, information channels, and distances).

(2) The parameters for the production, storage, and transportation policies at each location, including the structure of times and costs.

(3) The type of model used to represent customer demand (forecasting data, probability distributions, etc.).

(4) Data related to SC simulation (length of simulation horizon, time horizon for the plants scheduling models, etc.).

It also contains data related to the computation of the proposed scheme (state of the random number generator, number of iterations for the outer optimization loop and the Monte Carlo loop, termination criteria, etc.).

The agent-based framework helps in the generation and configuration of a simulation model by selection, instantiation, and composition of sets of components without the need for extensive programming expertise. The inclusion of uncertainty is easily managed as the simulator accepts the definition of parameters and the occurrence of events on the time line as belonging to probability distributions.

The simulator generates the input data for the scheduling model of the plant process, which is used periodically on production decisions. The scheduling model is formulated as a mixed-integer linear programming problem (MILP) and is based on the formulation presented by Kondili, Pantelides, and Sargent.[41] It is solved using GAMS−CPLEX[40] and MATLAB.[42] GAMS is interfaced with MATLAB, using the library developed by Ferris.[43]

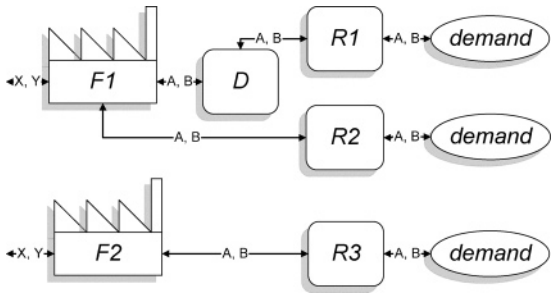

Figure 8. Schematic depiction of the SC network for the case study with 6 entities (case study I).

The objective function calculation module encloses data to calculate the simulation performance (objective function) from the simulation outputs and the .scm file. It is coded in C#.

The optimization module is a customized GA coded in MATLAB and wrapped as a dynamic link library (dll) to enable the communication with the agent system.

6. Case Study

6.1. Case Problem I. The SC scheme that is considered consists of six entities interconnected as shown in Figure 8. There are two plants, F1 and F2, and a distribution center, D, that manages materials and information to satisfy incoming demands from the three retailers—R1, R2, and R3, located in three sales regions. Raw materials X and Y enter the plants, and the manufactured products A and B are distributed by the remainder of the chain to the sales regions. Plant F1 supplies to sales region 2 and the distribution center (D), which, in turns, supplies sales region 1. On the other hand, plant F2 replenishes the retailer in sales region 3.

6.1.1. Storage Facilities. With regard to the inventory policy, a periodic revision strategy is implemented for all the inventories of products j at entities i, with τ_{ij} being the time period between two consecutive inventory reviews. Every τ_{ij} time units, the inventory position Inv_{ijt} is checked. If Inv_{ijt} is below the reorder point s_{ij}, a replenishment quantity $u_{ijt} = s_{ij} - \mathrm{Inv}_{ijt}$ is ordered; in other words, the replenishment quantity is proportional to the difference between the reorder point and the current inventory level. If the position is above s_{ij}, nothing is done until the next review. Thus, the strategy has two parameters whose values must be determined (τ_{ij} and s_{ij}) for all the inventories. In this case study, there are six inventory facilities: one in each site of the chain.

Furthermore, it is assumed that every inventory facility in Figure 8 has an inventory control policy with two parameters to tune (s_{ij} and τ_{ij}) for each product j. Thus, there are 24 variable values to determine.

The goal in this case study is to obtain the parameter values associated with the inventory control (e.g., τ_{ij}, s_{ij}) of products j at nodes i, at the beginning of the time horizon, in such a way that the expected SC operation profit over the time horizon is maximized.

Although any other operating parameter might be selected for optimization, the following case studies involves inventory parameters as decision variables, because of the interest they entail in pure pull systems. Quantitative issues related to inventory management are addressed in a large body of literature known as stochastic inventory theory. Based on foundational works such as the classical Economic Order Quantity (EOQ) model, research in inventory theory has been ongoing to extend these analytical results to the conditions experienced in the day-to-day SC operation.[44] However, a key limitation of all these

Table 1. Case Study I: Inventory Parameters before the Optimization Process[a]

	Raw Materials Warehouses				Distribution Center		Retailers					
	Plant1		Plant2		DC1		Ret1		Ret2		Ret3	
	$\tau_{i,j}$	$s_{i,j}$	$\tau_{i,j}$	$s_{i,j}$	$\tau_{i,j}$	$s_{i,j}$	$\tau_{i,j}$	$s_{i,j}$	$\tau_{i,j}$	$s_{i,j}$	$\tau_{i,j}$	$s_{i,j}$
raw materials												
X	25	56	24	44								
Y	13	56	21	44								
manufactured products												
A					13	46	15	47	8	44	6	55
B					12	49	23	56	8	40	19	55

[a] $[\tau_{i,j}] = h$, $[s_{i,j}] = u$.

studies is that the effects that production and distribution tasks may have on inventories are not considered.

6.1.2. Production Facilities. With respect to the batch plants embedded in the SC, two multipurpose batch chemical plants that can be described through the States-Tasks-Network (STN) representation[41] are considered. The following data are assumed to be known in advance:

(1) The set of raw materials and intermediate and final products to be manufactured and stored at the plant.

(2) The set of production recipes of final products.

(3) The amount of equipment, and their capacities and suitabilities, for the labor tasks.

(4) Economic information (prices of product and raw material, cost coefficients for utilities consumption and holding inventory over the horizon, etc.).

The mathematical formulation applied to compute the scheduling decisions associated with the batch plants has been taken from the work of Shah, Pantelides, and Sargent,[45] in which the authors reformulated the original assignment constraints used by Kondili, Pantelides, and Sargent.[41]

6.1.3. Objective Function. The optimality criterion used is the expected total reward $\mathscr{T}(\eta) = E[\text{Profit}(\eta,\omega)]$. Given some specific inventory control policy η, the total SC profit under a certain scenario ω (Profit(η,ω)) is calculated as the difference between the revenues ($R(\eta,\omega)$) and the global cost ($C(\eta,\omega)$):

$$\text{Profit}(\eta,\omega) = R(\eta,\omega) - C(\eta,\omega) \tag{3}$$

For the sake of simplicity, hereafter, η and ω have been dropped from the name of the variables. The dependence of the variables on η and ω then is not explicitly shown.

The revenues (R) are obtained according to eq 4:

$$R = \sum_t \sum_{j\in\text{Prod}} \sum_i Q_{ijt} \cdot \text{price}_{ij} \tag{4}$$

where Q_{ijt} is the quantity (in units) of final product j produced at node i at time t, and price_{ij} represents the unit price of j at node i. Prod is the subset of materials managed by the SC that are sold to external customers.

The global cost, C, is defined as

$$C = \text{TrnC} + \text{InvC} + \text{OrdC} + \text{PrdC} \tag{5}$$

In this expression, TrnC is the transport cost, InvC is the inventory cost, OrdC is the penalization for cumulated nonsatisfied orders, and PrdC is the production cost, in the time horizon [1, ...τ, ..., L].

The transport cost has two terms, as eq 6 shows:

$$\text{TrnC} = \sum_i \text{Ntrip}_i \cdot \text{ftc}_i + \sum_i \sum_j \text{Qtrip}_{ij} \cdot \text{vtc}_{ij} \tag{6}$$

where Ntrip_i is the number of trips from node i made within the time horizon; ftc_i is the fixed transportation cost per trip; Qtrip_{ij} is the transported quantity of product j (in units) from node i within the time horizon; and vtc_{ij} is the variable transportation cost per transported unit.

InvC is calculated as the sum of fixed and variable inventory costs:

$$\text{InvC} = \sum_t \sum_j \sum_i \text{fic}_{ijt} + \sum_t \sum_j \sum_i \text{Inv}_{ijt} \cdot \text{vic}_{ijt} \tag{7}$$

where fic_{ijt} is the fixed inventory cost of j at node i at time unit t; Inv_{ijt} is the stored quantity of j (in units) at node i per time unit t; and vic_{ij} is the variable inventory cost at node i per stored unit of j and per simulation step t.

The penalization for cumulated nonsatisfied orders is obtained according to eq 8:

$$\text{OrdC} = \sum_t \sum_j \sum_i \text{Ord}_{ijt} \cdot \text{oc}_{ijt} \tag{8}$$

where Ord_{ijt} is the orders quantity of j (in units) backlogged at node i and time t, and oc_{ijt} represents the cumulated orders cost at node i per unit of j and per simulation step t.

Finally, the production cost is obtained according to eq 9:

$$\text{PrdC} = \sum_t \sum_j \sum_i \text{Qprod}_{ijt} \cdot \text{vpc}_{ij} + \sum_t \sum_{j\in\text{ERM}} \sum_i \text{Qrm}_{ijt} \cdot \text{prm}_{ij} \tag{9}$$

where Qprod_{ijt} is the quantity of product j (in units) produced at node i at time t, and vpc_{ij} represents the unit variable production cost of j at node i; Qrm_{ijt} is the quantity (in units) of raw material j consumed at node i at time t, and prm_{ij} represents the price of raw material j at node i. ERM is the subset of materials circulating in the SC that are raw materials. Tables 1−8 show the data sets used in this work for the case study.

6.1.4. Demand. Demand is modeled as a set of discrete events distributed over the time horizon of the study, each of these events having an associated product, amount of material, and time of occurrence. The model considers that the material amounts are normally distributed around a mean value and the interarrival time intervals are uniformly distributed. The values of the stochastic parameters for a given simulation run are selected from the corresponding probabilistic distribution following the Monte Carlo way. For this specific case study, Figure 9 shows two samples of the demand according to the mean value and variance used in the case study (see Table 9).

6.2. Case Problem II. This is a more-complex case in terms of the size and the product exchange policy between the different nodes. It is presented with the purpose of demonstrating that the power of the approach is not limited to small examples.

Table 2. Case Study I: Fixed Inventory Costs at Each Node, Per Product and Simulation Step

	Fixed Inventory Cost, fic_{ijt} [$/u/t]							
	Raw Materials Warehouses		Finished Products Warehouses		Distribution Center	Retailers		
	Plant1	Plant2	Plant1	Plant2	DC1	Ret1	Ret2	Ret3
raw materials								
X	0.01	0.01						
Y	0.01	0.01						
manufactured products								
A			0.01	0.01	0.01	0.01	0.01	0.01
B			0.01	0.01	0.01	0.01	0.01	0.01

Table 3. Case Study I: Variable Inventory Costs at Each Node, Per Product and Simulation Step

	Variable Inventory Cost, vic_{ijt} [$\times 10^{-3}$ $/u/t]							
	Raw Materials Warehouses		Finished Products Warehouses		Distribution Center	Retailers		
	Plant1	Plant2	Plant1	Plant2	DC1	Ret1	Ret1	Ret3
raw materials								
X	2.0	2.0						
Y	2.0	2.0						
manufactured products								
A			0.7	1.0	0.5	1.0	1.5	1.0
B			0.7	1.0	0.5	1.0	2.5	1.0

Table 4. Case Study I: Cumulated Order Costs at Each Node, Per Unit Backlogged and Simulation Step

	Cumulated Order Cost, Oc_{ijt} [$\times 10^{-3}$ $/u/t]					
	Production Plants		Distribution Center	Retailers		
manufactured product	Plant1	Plant2	Dc1	Ret1	Ret2	Ret3
A	4.0	3.5	2.5	6.0	5.0	5.0
B	4.0	3.5	2.5	6.0	5.0	5.0

Table 5. Case Study I: Production Costs Per Unit Manufactured at Each Plant

manufactured product	Production Costs Per Unit Manufactured at Each Plant, vpc_{ij} [$/u]	
	Plant1	Plant2
A	1.0	1.0
B	1.0	1.0

Table 6. Case Study I: Raw Material and Product Prices

	Product Price [$/u]				
	Plant1	Plant2	sales region 1	sales region 2	sales region 3
raw materials					
X	0.5	0.5			
Y	0.5	0.5			
manufactured products					
A			7	7	11
B			8	8	13

Table 7. Case Study I: Fixed Transportation Cost Per Trip

Variable Transportation Cost per Trip, ftc_{ij} [$\times 10^{-2}$ $/u]			
From Plant1		From DC1	From Plant2 to DC1
to DC1	to Ret2	to Ret1	to Ret3
1.0	1.0	1.0	1.0

The SC layout considered in this case consists of twelve entities connected as shown in Figure 10. This SC network involves three manufacturing plants (Plant 1−Plant 3) that consume raw material X. One of them has its own supplier, S, and the others get X from external sources. The manufactured products (A, B, and C) are distributed to the markets through a network of five retailers (Ret1−Ret5) and three distribution centers (DC1−DC3).

Table 8. Case Study I: Variable Transportation Cost Per Unit

manufactured product	Variable Transportation Cost per Unit, vtc_{ij} [$\times 10^{-2}$ $/u]			
	From Plant1		From DC1	From Plant2
	to DC1	to Ret2	to Ret2	to Ret3
A	5.0	6.0	3.0	7.0
B	5.0	6.0	3.0	7.0

A periodic revision strategy similar to that described in Section 6.1.1 is implemented with the supplier, the plants, and the distribution centers. In this strategy, every τ_{ij} time units, the inventory position Inv_{ij} is checked. If it is below the reorder point s_{ij}, a replenishment quantity $u_{ij} = S_{ij} - Inv_{ij}$ is ordered to raise the stock level to S_{ij}. If the position is above s_{ij}, nothing is done until the next review. Therefore, this strategy has three parameters to determine for each node i and manipulated product j (τ_{ij}, S_{ij}, and s_{ij}). The number of decision variables is: 3 for the supplier, 36 for the plants and 27 for the distribution centers.

Retailers have a continuous inventory revision strategy. In this case, the revision is done every time material leaves the retailer; then, the parameters to be determined for retailers are S_{ij} and s_{ij} for each handled product and node (30 decision variables).

As in the simpler case, the manufacturing plants are modeled as multipurpose batch chemical plants and described using the STN representation. The objective function is the expectation

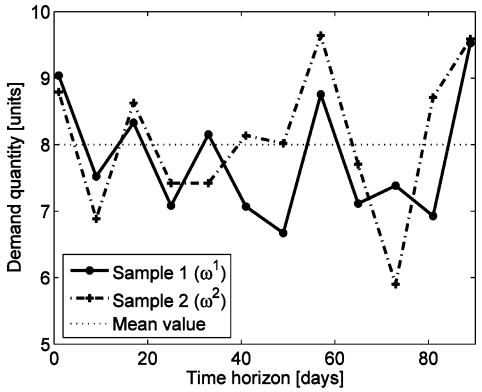

Figure 9. Case study I: Two sample path realizations for demand of product A at retailer Ret3.

Table 9. Case Study I: Demand Data at Each Sales Region

	Sales Region 1		Sales Region 2		Sales Region 3	
	product A	product B	product A	product B	product A	product B
mean value [u]	10	20	8	22	18	15
variance	1	1	1	1	1	1
frequency [day^{-1}]	0.78	0.19	0.31	0.33	0.56	0.78

on the total SC profit, as in Section 6.1. Demand is also considered as a set of events distributed over the time horizon of the study. The material amounts are normally distributed around a mean value and the interarrival time intervals are uniformly distributed.

The goal in this centralized approach is to obtain the parameter values associated with the inventory control policy for the 12 nodes and for each product manipulated at each node. The total number of variables involved is 96. Because of space limitations, data will be omitted. However they are available from the authors by request.

6.3. Results and Analysis. A key point in the proposed methodology is determination of the number of samples or replications (n) to take during the Monte Carlo simulation to ensure that E[Profit(η,ω)] is a good enough estimation for the mean of the universe of uncertain variables. As n gets larger, the representativeness of E[Profit(η,ω)] increases as guaranteed by the central limit theorem; however, the exact relationship must be studied, because it is dependent on the simulation model under consideration.

The specific objective of this analysis is to obtain an estimate of the mean with a relative error of $\gamma = 0.05$ and a confidence level of $100(1-\alpha)\%$ ($\alpha = 0.1$). The sequencial procedure then is as follows:

Step I. Make $n_0 \geq 2$ replications of the simulation and set $n = n_0$.

Step II. Let ω_l be a random variable defined on the lth replication for $l = 1, 2, ..., n$. Compute the arithmetic average $\bar{\omega}(n)$ and the confidence-interval half-length $\delta(n,\alpha)$ from ω_1, ω_2, ..., ω_n:

$$\delta(n,\alpha) = t_{n-1,(1-\alpha)/2}\sqrt{\frac{S^2(n)}{n}} \qquad (10)$$

$S^2(n)$ is the sample variance of ω_l and $t_{n-1,(1-\alpha)/2}$ is the critical point for the t-distribution.

Step III. If $\delta(n,\alpha)/|\bar{\omega}(n)| \leq \gamma/(1-\gamma)$, use $\bar{\omega}(n)$ as the point estimate for the mean and stop. Otherwise, replace n by $n+1$, make an additional replication of the simulation, and go to step II.

A more-detailed explanation about this procedure can be found in the work of Law and Kelton.[32]

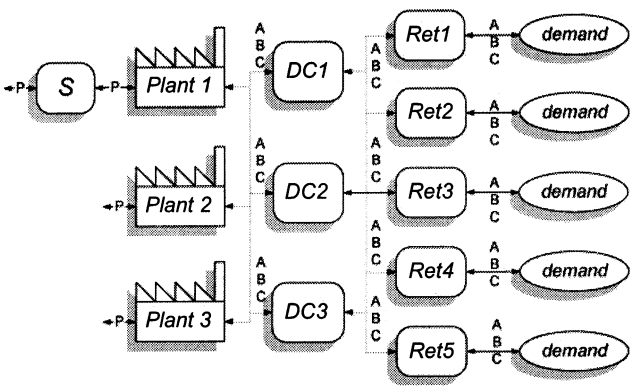

Figure 10. Schematic depiction of the SC network for the case study with 12 entities (case study II).

Table 10. Case Study I: Settings

parameter	value
number of decision variables of the outer loop	24
number of timelines, n	50
population size, N	10
maximum number of outer loop iterations, MaxGen	100
crossover probability, Px	0.9
mutation probability, Pm	0.05
entire strategy CPU time	28 h
timeline CPU time	0.7 s

The results of this study can be seen in Figure 11. After 45 simulations, the quotient $\delta(n,\alpha)/|\bar{\omega}(n)|$ falls below $\gamma/(1-\gamma) = 0.0526$ ($\gamma = 5\%$). It allows us to conclude that, in this case, a relatively small number of simulations is sufficient to obtain statistically significant performance estimates.

When a value of n needed to accomplish the confidence and error estimation requirements has been determined, the case problem has been solved with the proposed strategy.

In the first case study presented, the horizon of the planning model is 3 months, one simulation step accounts for 2.16 h, and replenishment and production decisions are made at the beginning of each period τ for each node in the SC network. Each chromosome in the GA coding consists of a string of all the decision variables. The influence of the population size (N) on the algorithm performance, as well as the crossover and mutation probabilities, have been studied. The termination criteria consists of reaching a prefixed maximum number of generations, MaxGen. In relation to this observation, there is a tradeoff between N and MaxGen, because the same level of convergence can be achieved by increasing both of these parameters. A relatively big MaxGen value has been determined, because of the increasing amount of data stored, which would demand an increase of N, thus saving virtual memory during the execution of the algorithm. The number of simulations within each Monte Carlo campaign is set to 50, and the number of generations in the outer loop is set to 100. The settings for the strategy in this case are summarized in Table 10. The computation time for the complete case has been assumed to be ~28 h on an AMD Athlon XP 2000 computer.

In regard to the scheduling problem to be solved at each plant, for instance, a model for a 5-day schedule corresponding to Plant1 with a discrete time mixed-integer linear programming

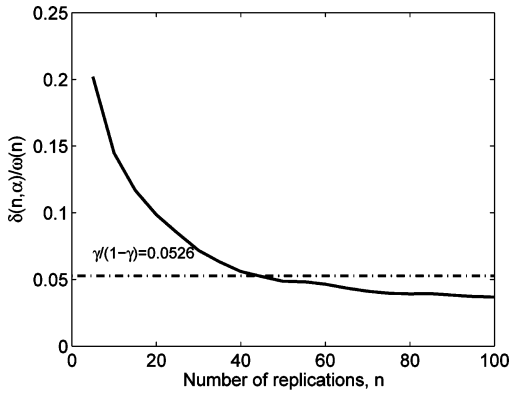

Figure 11. Case study I: Convergence study for the Monte Carlo simulation.

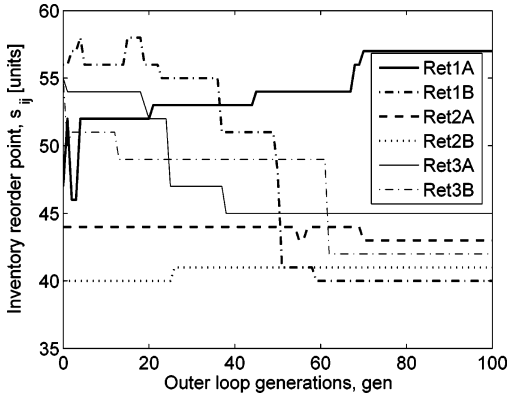

Figure 12. Case study I: Evolution of the reorder points s_{ij} for all the retailers.

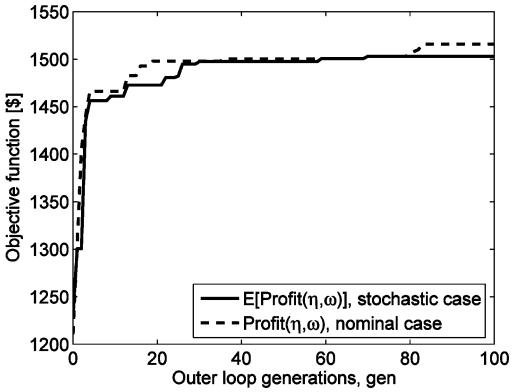

Figure 13. Case study I: Evolution of the objective function E[Profit-(η,ω)] during the optimization process.

(MILP) formulation entails ~6320 constraints and 6069 variables, among which 2420 variables are integer. The time required to obtain solutions with a 1% integrality gap on an AMD Athlon XP 2000 computer is in the range of 0.5−5 s, using GAMS−CPLEX, for the case problem that has been discussed. Given that the proposed approach requires a large number of simulations and, thus, a larger number of scheduling executions, the total computational load increases considerably.

Figure 12 illustrates the evolution of the inventory parameter s_{ij} at the three retailers, in the best-fitted individual of the population, as the outer loop iterations progressed. In the last iterations, variables almost do not change. In the same way, Figure 13 shows how the objective function E[Profit(η,ω)] evolves along with the number of outer loop iterations. In this case study, one strategy run results in an average improvement of 18%. In this figure, the evolution of Profit(η,ω) for the nominal case, i.e., using the strategy in Figure 2 but without using the Monte Carlo sampling loop (the deterministic case), also can be observed. The assessment of the obtained solution through all the scenarios used in the stochastic case gives a value of E[Profit(η,ω)] = \$1,453.05, 3.32% less than the stochastic solution.

Figures14 and 15 show the inventory levels that have been found at each of the nodes in the SC network for the initial and final solution. Generally, the inventory levels corresponding to the final solution are lower than the initial ones. Moreover, in the final solution, inventory shortages are avoided, because they never fall to zero in this case.

Figure 16 shows a Gantt chart that corresponds to one execution of the scheduled plan, at Plant1. Table 11 presents a summary of the results that concern the objective function and the corresponding parameter values for the node Ret1 in the SC network.

These results illustrate the effectiveness of the computational framework in optimizing the large number of variables from poor estimates in relatively few outer loop iterations.

Concerning the second case study considered, the number of simulation timelines required to obtain a good estimation for the expected profit, with a relative error of $\gamma = 0.05$ and a confidence level of $100(1 - \alpha) = 90\%$, was determined to be $n = 90$. The case problem then has also been solved with the

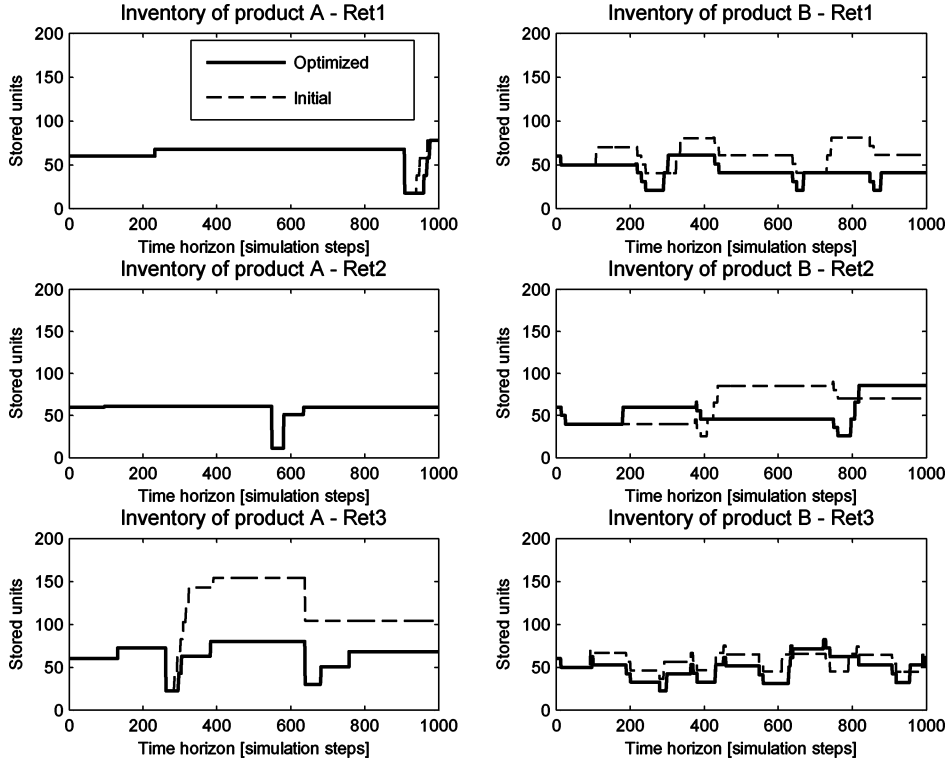

Figure 14. Case study I: Inventory levels in the SC network before and after the optimization process (data for retailers Ret1, Ret2, and Ret3).

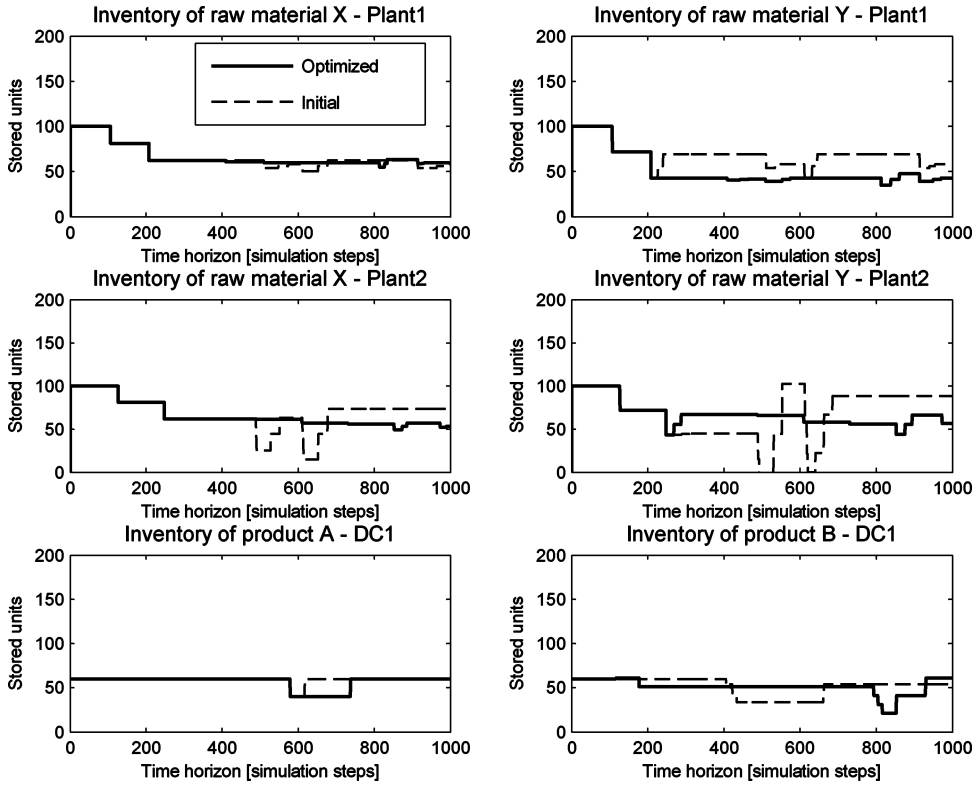

Figure 15. Case study I: Inventory levels in the SC network before and after the optimization process (data for production plants Plant1 and Plant2, and distribution center DC1).

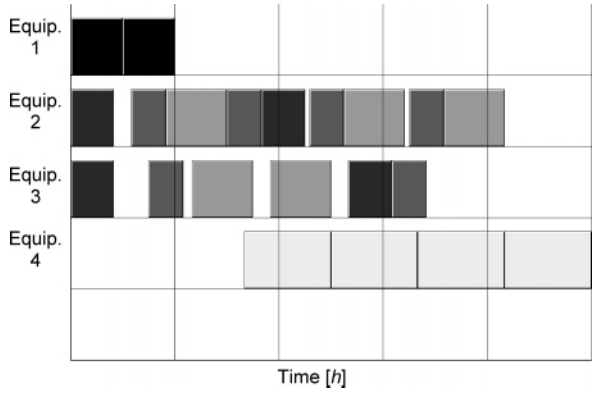

Figure 16. Case study I: Gantt chart for one schedule at Plant1.

Table 11. Case Study I: Summary of the Results at One Node of the Supply Chain (SC)

outer-loop iterations, gen	E[Profit(η,ω)] [$]	Retailer Ret1, Product A	
		s_{ij}	R_{ij}
0	1229.68	56	23
10	1461.00	56	13
20	1472.70	56	13
30	1497.42	55	13
40	1497.42	51	13
50	1497.42	48	13
60	1500.69	40	13
70	1502.89	40	13
80	1502.89	40	13
90	1502.89	40	13
100	1502.89	40	13

proposed strategy. The settings for the entire strategy and the results obtained are presented in Tables 12 and 13.

Figure 17 shows the evolution of the objective function through the generations in this case. The algorithm converges more slowly than in the previous case. Because of the complexity of the model, one simulation run requires ~2 s (0.7 s in the previous case). Therefore, the execution of the simulation-based optimization algorithm requires ~100 h (see Table 13) on an AMD Athlon XP 2000 computer.

Table 12. Case Study II: Settings

parameter	value
number of decision variables of the outer loop	96
number of timelines, n	90
population size, N	10
maximum number of outer loop iterations, MaxGen	200
crossover probability, Px	0.9
mutation probability, Pm	0.1
entire strategy CPU time	98 h
timeline CPU time	2.1 s

Table 13. Case Study II: Summary of the Results

outer-loop iterations, gen	E[Profit(η,ω)] [$]
0	2450.39
20	2925.72
40	2970.70
60	2970.70
80	2990.41
100	2990.41
120	3000.72
140	3010.52
160	3010.52
180	3010.52
200	3010.52

Unfortunately, let us mention that, even though the proposed framework is conceptually related to both approaches that have been previously described (stochastic programming and optimal control), in practice, it is very difficult to make comparisons with the results obtained using one or another of the approaches. That is mainly because of the different modeling principles used. Moreover, in the aforementioned approaches, the SC behavior is completely defined by a set of equations, wherea, in the proposed approach, the agent model is not equation-oriented.

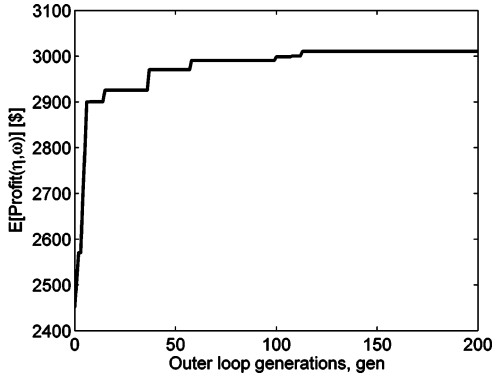

Figure 17. Case study II: Evolution of the objective function E[Profit(η,ω)] during the optimization process.

7. Conclusions

This work has considered the decision process aimed at improving the supply-chain (SC) operation under uncertain scenarios. Because of the huge complexity, both in the modeling and computation effort associated with solving this class of problems, an increasing amount of attention in the research community recently has been observed.

Taking into consideration the existing solution approaches and the underlying problem, a practical framework has been proposed to efficiently find decisions related to the operational/tactical level. The presented approach relies on the use of a hybrid strategy involving two nested loops. The inner loop entails the generation of different scenarios that are then simulated using a multi-agent simulator for the SC, whereas the outer loop involves an optimization process based on genetic algorithms (GAs).

In spite of the fact that the optimality of the solutions cannot be guaranteed, the approach is able to provide reasonable and practical solutions, using building blocks that are easy to understand and readily implemented in practice. The comparison with other approaches cannot be made only on the basis of reduction in the computational effort required to solve large instances of stochastic optimization problems. In fact, we believe that the framework described in this work can also lead to important savings in terms of the money and time spent in modeling complex SC systems, which would be hard to be represented by standard mathematical programming tools.

Finally, the problem can be extended in the future to include other decisions with different frequencies of updating, uncertainty sources at different levels, and multiple objective functions. In addition, the agent-based simulator can be improved by taking into account the negotiation with external suppliers and customers.

Acknowledgment

Financial support received from the Spanish "Ministerio de Educación y Ciencia" (FPU programs), "Generalitat de Catalunya" (FI programs), and GICASA-D (I0353), OCCASION (DPI2002-00856) and PRISM (MRTN-CT-2004-512233) projects is gratefully acknowledged. In addition, the authors would like to express their gratitude to Francisco Urbano for his support and assistance in developing this work.

Nomenclature

$C(\eta,\omega)$, C = total supply-chain (SC) cost in θ
$E[\cdot]$ = expected value
ERM = set of raw materials
fic_{ijt} = fixed inventory cost of product j at i and time t
ftc_i = fixed transportation cost from i per trip
$f(\eta,\omega)$ = performance index
$\mathcal{F}(\eta)$ = objective function
h = generic time period bigger than θ
H = number of time periods h
InvC = inventory cost in θ
Inv_{ijt} = inventory position at entity i and time t
L = number of time subperiods τ
MaxGen = maximum number of generations for the outer loop
n = number of Monte Carlo samples in the inner loop
N = number of individuals per population in the GAs
Ntrip_i = number of trips from node i in θ
oc_{ijt} = cumulated orders cost at i per unit of product j and t
OrdC = penalization for nonsatisfied orders in θ
Ord_{ijt} = orders for product j backlogged at node i and time t
PrdC = production cost in θ
prm_{ij} = price of raw material j at node i
price_{ij} = price of product j at node i
Prod = set of final products
Profit(η,ω) = total SC profit per decision strategy η and timeline ω
Q_{ijt} = units of final product j leaving node i at time t
$Q\text{prod}_{ijt}$ = units of product j produced at node i at time t
$Q\text{rm}_{ijt}$ = units of raw material j consumed at node i at time t
$Q\text{trip}_{ij}$ = units of j transported from node i in θ
$R(\eta,\omega)$, R = total SC revenues in θ
s_{ij} = inventory reorder point at entity i for product j
S_{ij} = inventory capacity of entity i for product j
$S^2(n)$ = variance of a sample of n values of ω_l
T = number of time periods θ
TrnC = transportation cost in θ
u_{ijt} = inventory replenishment quantity at entity i and time t
vic_{ijt} = variable inventory cost at node i per unit of product j and time t
vpc_{ijt} = variable production cost at i per unit of j and time t
vtc_{ij} = variable transportation cost from i per unit of product j

Greek Characters

α = one minus the desired confidence interval
γ = relative error of an estimate to the mean value
$\delta(n,\alpha)$ = confidence interval for n replications and confidence level of $100(1-\alpha)$ (%)
η, η^k_{gen} = set of generic decision variables (decision strategy)
θ = time period between the inventory parameters updating
τ, τ_{ij} = time sub-period between replenishment/production decisions
ω, ω_l = generic random variable realization
$\bar{\omega}(n)$ = arithmetic average on n samples of variable ω_l

Subscripts and Superscripts

gen = counter for the outer loop iterations
i = SC entity
j = SC product
k = each of the individuals or chromosomes in a population
l = replication in the Monte Carlo sampling
t = simulation step

Acronyms

CPI = chemical process industry
GA = genetic algorithm
MILP = mixed-integer linear programming
MINLP = mixed-integer nonlinear programming
NLP = nonlinear programming
RFQ = request for quote

SC = supply chain
SCM = supply chain management
STN = states and tasks network

Literature Cited

(1) Subramanian, D.; Pekny, J. F.; Reklaitis, G. V. A simulation−optimization framework for addressing combinatorial and stochastic aspects of an R&D pipeline management problem. *Comput. Chem. Eng.* **2000**, *24*, 1005−1011.

(2) Kafoglis, C. C. Maximize competitive advantage with a supply chain vision. *Hydrocarbon Process.* **1999**, *2*, 47−50.

(3) Han, C.; Douglas, J. M.; Stephanopoulos, G. Agent-based approach to a design support system for the synthesis of continuous chemical processes. *Comput. Chem. Eng.* **1995**, *19*, S63−S69.

(4) Maguire, P. Z.; Scott, D. M.; Paterson, W. R.; Struthers, A. Development of an advanced modelling environment. *Comput. Chem. Eng*. **1995**, *19*, S265−S270.

(5) Maguire, P. Z.; Struthers, A.; Scott, D. M.; Paterson, W. R. The use of agents both to represent and to implement process engineering models. *Comput. Chem. Eng.* **1996**, *20*, S571−S578.

(6) Siirola, J. D.; Hauan, S.; Westerberg, A. W. Toward agent-based process system engineering: proposed framework and application to nonconvex optimization. *Comput. Chem. Eng*. **2003**, *27*, 1801−1811.

(7) Siirola, J. D.; Hauan, S.; Westerberg, A. W. Computing Pareto fronts using distributed agents. *Comput. Chem. Eng.* **2004**, *29*, 113−126.

(8) Eo, S. Y.; Chang, T. S.; Shin, D.; Yoon, E. S. Cooperative solving in diagnostic agents for chemical processes. *Comput. Chem. Eng*. **2000**, *24*, 729−734.

(9) Pendharkar, P. C. A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling systems. *Expert Syst. Appl.* **1999**, *16*, 121−133.

(10) Knotts, G.; Dror, M.; Hartman, B. C. Agent-based project scheduling. *IIE Trans.* **2000**, *32*, 387−401.

(11) McGreavy, C.; Wang, X. Z.; Lu, M. L.; Zhang, M.; Yang, S. H. Objects, agents and work flow modelling for concurrent engineering process design. *Comput. Chem. Eng.* **1996**, *20*, S1167−S1172.

(12) Batres, R.; Lu, M. L.; Naka, Y. An agent-based environment for operational design. *Comput. Chem. Eng*. **1997**, *21*, S71−S76.

(13) Batres, R.; Asprey, S. P.; Fuchino, T.; Naka, Y. A KQML multi-agent environment for concurrent process engineering. *Comput. Chem. Eng*. **1999**, *23*, S653−S656.

(14) Parunak, H. V.; Savit, R.; Riolo, R. L.; Clark, S. J. DASCh: Dynamic analysis of supply chains, DASCh Final Report CEC-0 115 1999. Industrial Technology Institute, Ann Arbor, MI, 1999 (http://www.erim.org/~vparunak/dasch99.pdf, accessed January 30, 2006).

(15) Gjerdrum, J.; Shah, N.; Papageorgiou, L. G. A combined optimisation and agent-based approach for supply chain modelling and performance assessment. *Prod. Plann. Control* **2000**, *12*, 81−88.

(16) García-Flores, R.; Wang, X. Z. A multi-agent system for chemical supply chain simulation and management support. *OR Spectrum* **2002**, *24*, 343−370.

(17) Forrester, J. W. *Industrial Dynamics*; MIT Press: Cambridge, MA, 1961.

(18) Goodwin, R.; Keskinocak, P.; Murthy, S.; Wu, F.; Akkiraju, R. Intelligent decision support for the e-supply chain. In *Artificial Intelligence for Electronic Commerce, AAAI Workshop 99*, Orlando, FL, 1999; pp 77−81.

(19) Sauter, J. A.; Parunak, H. V. D.; Goic, J. ANTS in the supply chain. Presented at the *Workshop on Agents for Electronic Commerce at Agents '99*, Seattle, WA, 1999.

(20) Fox, M. S.; Barbuceanu, M.; Teigen, R. Agent-oriented supply chain management. *Int. J. Flexible Manuf. Syst.* **2000**, *12*, 165−188.

(21) Swaminathan, J. M.; Smith, S. F.; Zadeh, N. M. Modelling supply chain dynamics: a multi-agent approach. *Decis. Sci.* **1998**, *29*, 607−632.

(22) Julka, N.; Srinivasan, R.; Karimi, I. Agent-Based Supply Chain Management−1: Framework, *Comput. Chem. Eng.* **2002**, *26*, 1771−1781.

(23) Mele, F. D.; Guillén, G.; Urbano, F.; Espuña, A.; Puigjaner, L. A novel agent-based approach for supply chain retrofitting. In *Sixth International Conference on Foundations Of Computer-Aided Process Design*; Floudas, C. A., Agrawal, R., Eds.; Omnipress: Madison, WI, 2004; pp 461−464.

(24) Lee, Y. H.; Cho, M. K.; Kim, S. J.; Kim, Y. B. Supply chain simulation with discrete-continuous combined modeling. *Comput. Ind. Eng.* **2002**, *43*, 375−392.

(25) Biswas, S.; Narahari, Y. Object oriented modeling and decision support for supply chains. *Eur. J. Operat. Res.* **2004**, *153*, 704−726.

(26) Birge, Z.; Louveaux, S. *Principles on Stochastic Programming*; Springer−Verlag: New York, 1997.

(27) Balasubramanian, J.; Grossmann, I. E. Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty. *Ind. Eng. Chem. Res.* **2004**, *43*, 3695−3713.

(28) Cheng, L.; Subrahmanian, E.; Westerberg, A. W. Design and planning under uncertainty: Issues on problem formulations and solutions. *Comput. Chem. Eng.* **2003**, *27*, 781−801.

(29) Cheng, L.; Subrahmanian, E.; Westerberg, A. W. Multiobjective decisions on capacity planning and inventory control. *Ind. Eng. Chem. Res.* **2004**, *43*, 2192−2208.

(30) Cheng, L.; Subrahmanian, E.; Westerberg, A. W. A comparison of optimal control and stochastic programming from a formulation and computation perspective. *Comput. Chem. Eng.* **2004**, *29*, 149−164.

(31) Jung, J. Y.; Blau, G.; Pekny, J. F.; Reklaitis, G. V.; Eversdyk, D. A simulation based optimization approach to supply chain management under demand uncertainty. *Comput. Chem. Eng*. **2004**, *28*, 2087−2106.

(32) Law, A. M.; Kelton, W. D. *Simulation Modeling & Analysis*, 3rd Edition; McGraw−Hill: New York, 2000.

(33) Fu, M. C. Optimization for simulation: Theory vs. practice. *INFORMS J. Comput.* **2002**, *14*, 192−215.

(34) Andradóttir, S. A review of simulation optimization techniques. In *Proceedings of the Winter Simulation Conference*; Medeiros, D. J., Watson, E. F., Carson, J. S., Manivannan, M. S., Eds.; Washington, DC, 1998; pp 151−158.

(35) Syarif, A.; Yun, Y.; Gen, M. Study of multi-stage logistic chain network: a spanning tree-based algorithm approach. *Comput. Ind. Eng.* **2002**, *43*, 299−314.

(36) Vergara, F. E.; Khouja, M.; Michalewicz, Z. An evolutionary algorithm for optimizing material flow in supply chains. *Comput. Ind. Eng.* **2002**, *43*, 407−421.

(37) Zhou, G.; Min, H.; Gen, M. The balanced allocation of customers to multiple distribution centers in the supply chain network: a genetic algorithm approach. *Comput. Ind. Eng.* **2002**, *43*, 251−261.

(38) Banks, J. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*; Wiley: Hoboken, NJ, 1998.

(39) Wooldridge, M.; Jennings, N. Intelligent agents: theory and practice. *Knowl. Eng. Rev.* **1995**, *10*, 115−152.

(40) Brooke, A.; Kendrik, D.; Meeraus, A.; Raman, R.; Rosenthal, R. E. *GAMS−A User's Guide*; GAMS Development Corporation: Washington, DC, 1998.

(41) Kondili, E.; Pantelides, C. C.; Sargent, R. W. H. A general algorithm for shortOLINIT-term scheduling of batch operations. I. MILP formulation. *Comput. Chem. Eng.* **1993**, *17*, 211−227.

(42) *MATLAB 7.0 (R14), Simulink 6.0, Stateflow 6.0. User's Manual*; The MathWorks, Inc.: Natick, MA, 2004.

(43) Ferris, M. C. MATLAB and GAMS: Interfacing Optimization and Visualization Software, Computer Sciences Department, University of Wisconsin at Madison, Madison, WI, 2005 (<http://www.cs.wisc.edu/math-prog/matlab.html>, accessed January 30, 2006).

(44) Tayur, S.; Ganeshan, R.; Magazine, M. *Quantitative Models for Supply Chain Management*; Kluwer Academic Publishers: Cambridge, U.K., 1998.

(45) Shah, N.; Pantelides, C. C.; Sargent, R. W. H. A general algorithm for shortOLINIT-term scheduling of batch operations. II. Computational issues. *Comput. Chem. Eng.* **1993**, *17*, 229−244.

Received for review October 7, 2005
Revised manuscript received February 8, 2006
Accepted February 16, 2006

IE051121G