The Simulation-based Multi-objective Evolutionary Optimization (SIMEON) Framework

Ronald Apriliyanto Halim
Systems Engineering Group,
Faculty of Technology, Policy and Management
Delft University of Technology
Jaffalaan, 5, 2628BX,
Delft, THE NETHERLANDS
31 (0)15 27 88380
R.A.Halim-1@student.tudelft.nl

Mamadou Diouf Seck
Systems Engineering Group,
Faculty of Technology, Policy and Management
Delft University of Technology
Jaffalaan, 5, 2628BX,
Delft, THE NETHERLANDS
31 (0)15 27 88380
M.D.Seck@tudelft.nl

ABSTRACT
The combination of simulation and optimization has been successfully applied to solve real-world decision making problems. However, there is no formal structure to define the integration between simulation and optimization. This consequently deters the development of simulation-based optimization methods that have a proper balance between the desired features (i.e. generality, efficiency, high-dimensionality and transparency). This research provides two contributions to the problem above by providing: 1) the design of the framework that facilitates the fulfillment of the aforementioned features; 2) the implementation of the framework in Java. The proposed framework is developed based on Zeigler’s modeling and simulation framework and the phases of an optimization study in operations research. The test and evaluation show that the desired features are successfully satisfied.

Categories and Subject Descriptors
I.6.5 [Model Development]: Simulation-based multi-objective evolutionary optimization framework: simulation, optimization, modeling and simulation framework.

General Terms
Algorithms, Performance, Design, Economics, Experimentation, Theory

Keywords
framework; simulation; multi-objective; evolutionary optimization; NSGAII.

1. INTRODUCTION
In the real world, decision making often involves a trade-off between the satisfaction of multiple conflicting objectives. This fact can be seen in different problem domains, covering both social and technological systems, where a single or multiple parties are involved [1, 7, 8, 15, 30, 31]. Many studies in the field of decision sciences/operations research have been performed to support decision makers in formulating acceptable solutions to this problem [25]. Among those studies, simulation and optimization turned out to be prominent approaches that are widely used to assist decision making process [7, 9, 25]. Furthermore, successful efforts to combine simulation and optimization methods have been reported in many researches that aim at solving real-world multi-objective optimization problems. This nurtures the growth of researches in so-called simulation-based optimization field [2, 14, 19]. In the simulation-based optimization method, optimization typically functions as the search method that explores the solution space in such a way that solutions leading to the preferred system performance(s), assessed by a simulation model, can be found. Unlike the traditional optimization methods, which use a mathematical model, the simulation-based optimization method allows an accurate representation of the dynamics and stochastic nature of the real system. Furthermore, the best solutions can be found without the tedious effort of manually traversing the whole possible decision alternatives.

However, there is a knowledge gap in this widely applied method. There has not been any formal and detailed explanation on the way simulation and optimization techniques should be integrated. Often, a simple conceptualization of either the simulation or the optimization components are presented as black boxes ([2];[5]; [26, 36];). This knowledge gap thus, might deter the effective use of optimization techniques by simulation practitioners and vice versa.

Furthermore, other challenges that result from the lack of a structured approach can also be found by looking at the conflicts found in the fulfillment of the broader requirements formulated by Fu in [17]. These requirements include generality, transparency to user, high dimensionality and efficiency. Among those requirements, generality very often conflicts with efficiency [17, 18]. This can also be seen from the optimization routines that are being developed in this field. Most of them employ evolutionary-based algorithms, which are designed in tight coupling with the modeled-problem to allow efficient exploitation of the problem structure. This tight coupling sacrifices the generality of the method, which would actually allow solving a wide range of problems across different domains [2, 14, 15, 23, 31]. While usability can often be facilitated with a user friendly user interface, there are many shortcomings pertaining to efficiency and high dimensionality in the simulation-based optimization commercial packages. A review of two popular optimization routines in commercial simulation software (AutoStat and OptQuest) reveals that both packages lack an efficient multi-objective optimization routine [17]. AutoStat approaches the
multi-objective optimization problem using the classical method in which multiple objectives are aggregated to form a single objective using a weight-vector. The drawback of this method is that for different preferences of the decision maker(s), different weight-vectors have to be used and the same problem has to be solved repetitively [34]. The similar approach was also employed by OptQuest, which only started to provide an efficient multi-objective optimization routine, allowing the analysis of the Pareto Frontier, after the release of engine v6.5 (http://www.opttek.com/Products/Documentation.html).

The framework of modeling and simulation proposed by Zeigler in [37], due to its clear characterization of the relevant concepts, is a valuable starting point for tackling the problems mentioned above. In this framework, the principle of separation of concerns is well applied. Firstly, there is a separation between the model and the context under which it is experimented with. Furthermore, there is a separation between the simulator and the simulation model. This framework is therefore a good basis for the detailed formalization of the structural relationships between simulation and optimization techniques. More importantly, the resulting framework can also be used to ensure the proper balance in realizing the various desired features mentioned above.

The proposed framework will make two contributions. The first is to provide a transparent structure and the formal definitions of the simulation-based optimization method. These structure and definitions will contribute to the development of a generic simulation-based optimization methodology. The second contribution is to provide a detailed design and a Java implementation of simulation-based optimization method which is able to fulfill the desired features mentioned by Fu in [17].

The rest of the section is organized as follows. Section 2 delineates the relevant theories for the development of a simulation-based multi-objective optimization framework. This is then followed by the detailing of the requirements and the conceptualization of the proposed framework in section 3. Next, section 4 presents the translation of the conceptualization into the system architecture and the detailed designs of the framework. In section 5, the results of the tests conducted to the framework are presented. Finally, section 6 presents the conclusion and the recommendation for the future work.

2. SIMULATION-BASED OPTIMIZATION METHODOLOGY

In this section the relevant work both from simulation and optimization fields are presented to provide the theoretical foundations with which a simulation-based multi-objective optimization framework can be built. We first present the detailed component of modeling and simulation framework presented by Zeigler in section 2.1. Subsequently, the description of multi-objective optimization problem and algorithm is presented in section 2.2.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

2.1 Experimental Frame (EF) in the Zeigler’s modeling and simulation framework

A relevant concept in Zeigler’s framework is the experimental frame which explains the separation of concerns between the model and any data gathering (e.g. statistical measurements), and any control efforts (e.g. starting and stopping of the simulation) that are not performed in the real system. To serve these functions, the concept of experimental frame is formalized with three components which govern the experimentation on a model: the generator, transducer and acceptor [4, 35]. The generator component produces the input segments to the simulation model. Those input segments are then processed by the model using certain state-transition functions to calculate all the resulting state changes. Next, the transducer maps the output variables into outcome measures of interest. Finally the acceptor component monitors the validity of the experiments by checking whether the values of the outcome measures of interest violate the pre-defined constraints of the experimentation.

Furthermore, in a simulation-based optimization method, experimentations can be performed to evaluate the effect of certain alternatives defined by the model builder in respect to certain pre-defined goals. In this case, the simulation model is conditioned to a certain set of circumstances that is defined in a so-called treatment [9]. Thus, in an experiment, a specific treatment is given to the simulation model, and the model is run to produce the outcome measures of interest through the acceptor (Figure 1). In the effort of optimizing the system performances, normally a number of treatments that involve different collections of input data are specified and executed in the experiments. However, this can be very challenging and sometimes impossible to do manually if the model is too complex or the relationship between the input data (or often called decision variables in optimization study) and the outcomes measure of interest is not understandable.

Therefore, we can see that a computer-based approach that implements an optimization technique is needed to perform the search of the optimal values of the decision variables in respect to the pre-defined goals. The next section shall delineate the formal structure of the problem encountered here multi-objective optimization problem. In addition to that, a state-of the art multi-objective optimization technique is also going to be presented.
2.2 Multi-objective optimization methodology

One of the desired features for a simulation-based optimization method is high-dimensionality. This feature mainly comes from the need to solve multi-objective optimization problems. Following section describes the formal structure of such problems and state-of-the-art algorithms to solve such problems.

2.2.1 Multi-objective optimization problem

Following is the formulation of a multi-objective optimization problem:

Minimize \[ z_i = f_i(x), \quad i = 1, 2, \ldots, m \] (1)

Subject to \[ g_i(x) \leq 0, \quad i = 1, 2, \ldots, m \] (2)

Note that \( f(x) \) is the objective function, where \( x \in \mathbb{R}^n \) is a vector of \( n \) decision variables, and \( g_i(x) \) are inequality constraint that consists of \( m \) functions that shape the feasible area.

Fundamentally, multi-objective optimization problem is different from single objective optimization problem. In single objective optimization, the search process is focused on finding one best solution that is superior to all other solutions. In the case where there are multiple objectives to optimize, it is not always the case that there exists a solution that is optimal in terms of all the objectives due to incommensurability and conflict among the objectives [22]. In case there is conflict among the objectives, a solution that is optimum in one objective may be the worst for the other objectives. In this condition, there is normally a set of solutions that cannot be compared with each other without adding additional information (such as preference structure upon the objectives). These kinds of solutions are normally regarded as non-dominated solutions or Pareto optimal solutions [29]. Non-dominated solutions have the characteristic that their optimality cannot be improved further without sacrificing at least one of the other objective functions.

2.2.2 Multi-objective evolutionary optimization algorithms

Ever since Evolutionary Algorithms (EAs) had been introduced to solve MOOP, its researches, use and popularity had increased rapidly and significantly over the past decade opening a rapidly growing research field namely Multi-Objective Evolutionary Algorithm [10]. The main advantage of this method in comparison with the classical methods is that it uses a population-based approach in which multiple solutions are simultaneously generated in each of its iteration. This gives MOEA better capability to explore larger criterion space in shorter computational time. Some of the well-known properties of MOEA that motivate the use of this method are:

1. MOEAs have the flexibility to be adapted to different problem structures and thus have wide application fields
2. MOEAs do not require gradient or derivative values such as what is needed by gradient-based algorithms
3. The decision maker(s) does not need to have an a priori articulation of preferences regarding the accomplishment of all the objectives before the solutions/alternatives are presented

Among, many MOEAs, the framework proposed will use Non-Dominated Sorting Genetic Algorithm II as its optimization engine. It is one of the state-of-the-art methods to solve MOOP [12]. Like any other EAs this algorithm also finds its root in evolutionary theory and is built on top of classic Genetic Algorithm. Despite many improvements and additional features that NSGAII has, the fundamental theory that underlies the adaptive capability of this algorithm is the same as that used in standard genetic algorithms. However, there are some features that distinguish NSGA-II from other EAs:

1. It employs elitism principle
2. It has explicit diversity preserving mechanism
3. It is focused to find the non-dominated solutions

Finally, since the framework uses evolutionary algorithm, we henceforth name it as the Simulation-based Multi-objective Evolutionary Optimization (SIMEON) framework.

3. SIMEON DEFINITION

To ensure that the conceptualization of SIMEON takes into account all the desired features formulated by Fu in [17], we firstly present the technical requirements elicited based on the real-world needs for the framework in section 3.1. Next, in section 3.2 the conceptualization of SIMEON will be presented based on these requirements.

3.1 SIMEON requirements

Fu in [17] has formulated four high level features that are desirable in a simulation-based optimization method. In this section we further detail those requirements based on the interviews to the potential users of SIMEON in the real-world. Note that we adapt the transparency requirement into the broader notion of usability proposed by Keen and Solin [28] to better fit this requirement into the context of the development of a user-friendly tool.

3.1.1 Generality

Among all the desired features mentioned by Fu in [17], generality is normally of the highest value for both scientific and business worlds as it enables practical implementation of the optimization methods to different simulation problems [17, 18]. Consequently, this makes a framework that employs problem-dependent optimization techniques becomes less desirable. Since SIMEON is going to be applied to different problem domains, the optimizer used should be designed not to be problem-structure dependent.

3.1.2 Efficiency

It is important to guarantee the convergence of the optimizer in a reasonable amount of computational time. This is because the framework will be implemented and used in a common office notebook. However, the requirement on efficiency is inevitably affected adversely by the requirement of generality. One consequence of treating the simulation model as a black box (as what is done by metaheuristic) is that the optimizer does not make use the information regarding the problem structure (e.g. gradient information) to solve the optimization problems [17]. This leads to a relatively slow performance in comparison to the optimizers that use such information to solve the same problem. However, this condition also can be mitigated by ensuring the extendibility...
of SIMEON with algorithms that could make use the information regarding the structure of the simulation model such as the perturbation analysis [20], weak derivatives [32], etc.

### 3.1.3 High-dimensionality

It is a requirement that gives SIMEON a distinction in comparison to the other simulation-based optimization framework. This requirement mainly comes from the need to optimize multi-objective optimization problems which are ubiquitous in the real world problems. Furthermore, there are also needs that come from the modeling perspective, they are:

1. The need to optimize quantitative or continuous or real-valued variables as this need still dominates big number of optimization problems [1, 15].
2. The need to optimize qualitative or discrete or integer variables which can be used to represent non-quantitative variables including structural alternatives for a simulation model [3, 33].
3. The need to optimize both qualitative and quantitative variables simultaneously [14].

### 3.1.4 Usability

It is important to distinguish and define the users of SIMEON. The first type of user is defined as non-expert users, i.e. those who have at least the basic knowledge of modeling and simulation techniques. These non-expert users are expected to be able to develop simulation models in various simulation packages and use them to carry out an analysis by setting up experiments. On the other hand, the expert users are expected to be the simulation experts who have knowledge level and skills that enable them to develop object-oriented simulation models and use them to solve different kinds of problems.

Another additional requirement would be that both the expert and non-expert users should have the basic knowledge of optimization which enables them to apply optimization techniques appropriately to solve different decision problems.

To operationalize the concepts mentioned above, here we present the definitions of the entities that are necessary to support a functional simulation-based optimization model.

#### a. Treatment

Treatment can be defined as a specific set of conditions that are imposed to the model to perform the simulation [9]. Those conditions cover:

1. Specification of input data
2. Collection of input data
3. Initialization conditions
4. Run control conditions, which include length of the warm-up period, run length, and number of replications
5. Specification of output data

In the optimization context, what normally distinguishes one treatment from the others is the value of the decision variables inside those treatments. Thus, in this case a treatment contains a specific set of decision variable values which set the initial states of the simulation model.

#### b. Experimental frame

The role of the components inside the experimental frame, following the definitions of transducer, generator and acceptor are given:

1. **The generator**
   
   This component produces the input segments to the simulation model. In the context of simulation based optimization, generator produces the values of variables other than the decision variables which are normally caused by events that take place in the environmental condition of the system (e.g. the arrival of entities into the system that follow certain statistical distribution).

2. **The transducer**
   
   This component maps the output variables into outcome measures of interest. These outcome measures are normally the statistical summaries of the simulation output which can be defined as the goals of system or objective functions of the optimization problem.

3. **The acceptor.**
   
   This component monitors the validity of the experiments by checking whether the outcome measures of interest violate the pre-defined constraints of the experimentation. Acceptor does not always exist in an experimental frame, and in this case, transducer can directly produce the outcome measures of interest.
c. Simulation-based optimization experiment
A simulation-based optimization experiment is an execution of the simulation model with a specific treatment and environmental conditions to produce the outcome measures of interest. Therefore, this type of experiment includes the specification of Zeigler’s experimental frame.

d. Simulation-based optimization problem
A simulation-based optimization problem is a form of optimization problem in which simulation experiments are used to find the optimal values for the decision variables. Thus, simulation-based experiments should be defined within such a problem. Moreover, the specification of this problem should distinguish the decision variables and the other variables (environmental variables) within the simulation model.

3. Develop a computer-based procedure for deriving solutions to the problem from the model.

The computer-based procedure that is going to be developed is the NSGAII. The jMetal library [16] will be used to develop further the NSGA-II along with the necessary operators and functionality. On the other hand, the DSOL library [27] will be used as the basis for the development of the simulation models that are going to be used for the test cases. In addition to that, to connect the algorithm to the simulation model and the experimental frame a procedure has to be developed as well.

Using the routines of NSGAII, the aim of SIMEON is to provide a mechanism that enables automatic and iterative specification of the values of the decision variables in such a way they are optimized in regards to the pre-defined objective functions. This means that the knowledge of the modeler to specify the treatments as depicted in

Figure 1 can be replaced by the routine of the NSGAII to find the optimal solutions for the problem. Finally, Figure 2 depicts the conceptual diagram of the SIMEON framework.

Based on the conceptual design above, we can see the separation of concerns between the optimization algorithm and simulation-based problem. This feature is important for the fulfillment of the generality requirement. Next, the requirement regarding efficiency is also covered in the implementation of the NSGA-II which is well-known to be fast enough to find the non-dominated solutions for multi-objective optimization problems. In addition to that, the high-dimensionality requirement can also be fulfilled by designing specific representations and operators for this algorithm based on the types of the decision variables to optimize (this will be elaborated further in section 4.2.2).

SIMEON is currently specialized for the integration of NSGA-II into Zeigler’s modeling and simulation framework. It is obviously possible to integrate other evolutionary-based optimization techniques.

4. SIMEON DESIGN AND DEVELOPMENT
In this section, we present the system architecture and the detailed design of the SIMEON framework. The architecture is constructed based on the conceptual design presented in the previous section. It gives an accurate overview of the structure of the SIMEON framework. Furthermore, the detailed design will substantiate how we implement this architecture in an object-oriented language such as Java.

4.1 SIMEON system architecture
As SIMEON is a framework that is going to be used to develop decision support tools, its architecture is synthesized in such a way that it is coherent with the standard decision support system’s architecture proposed in [6], This is done to retain all the features and functionalities that a simulation-based decision support tool might have and add additional functionalities/services on top of it. SIMEON also uses a multi-tier architecture (Figure 3).

![Figure 2 Conceptual design of the SIMEON framework](image)

![Figure 3 System architecture of SIMEON](image)
As illustrated by the figure above, the architecture has three layers:

1. The presentation layer: a layer where the language and presentation systems are positioned. This layer serves mainly as the user interface though which the user interacts with SIMEON. There are several interfaces that take the messages from the user into SIMEON, serving as the language systems:
   a. The spreadsheet that is used to define the decision variables.
   b. The Java-based modeling interface, used to define the optimization model to be optimized. This frame takes as inputs, the output objects that are defined in the simulation model. This way, there is a separation of concern between modeling the outputs of the simulation model and modeling the optimization problem based on those outputs.
   c. The tab in the Java-based control panel where the users specify the algorithm to use, its parameters, the simulation model to use and its run control conditions.

On the other hand, there are also interfaces that emit messages to the user, serving as the presentation systems:
   a. The 2D plot and the spider chart that project the non-dominated solutions for multi-objective functions.
   b. The excel sheet that records the output of each framework execution.

2. The Problem Processing layer: a layer where the computation engines are positioned to provide the problem solving services. There are two distinctive applications/engines in this layer: the optimization and simulation engines. The optimization engine consists of multi-objective evolutionary algorithm and simulation-based optimization problem objects. Next, the simulation engine consists of the simulation model, the treatment that contains the decision variables for the model, and the simulation-based optimization experiment objects. It is important to note that there are multiple treatments and simulation-based optimization experiments in a framework execution. This is done to perform the multiple function evaluations (by means of simulation) based on the sets of decision variables specified in a set of treatments. Thus, one function evaluation that is specified within a simulation based problem object, involves the construction of one experiment with a specific treatment or a set of certain decision variables.

3. Knowledge/data layer: a layer where the knowledge system is positioned. This is the layer where all relevant knowledge and information needed by SIMEON are stored. An example of a knowledge system can be independent database systems (e.g., Microsoft Excel, Access, ERP, MySQL, etc) that contain data needed for the simulation or a configuration file for SIMEON.

4.2 SIMEON detailed design

In this section the detailed designs of the SIMEON sub-systems are presented. We first present the implementation structure of the framework in an object oriented language. Next, the design of genetic representations and operators used within SIMEON are instantiated. Last but not least, the features of the user interface are also going to be presented.

4.2.1 Design of the SIMEON framework

As noted in the system architecture diagram (Figure 3), there are six main components of the simulation and optimization engines. We implement these components as Java based objects.

The SimOptNSGAIII implements the main optimization routine, which solves a SimulationBasedProblem. The latter defines the SimBasedOptExperiment, which uses the Simulator, Treatment and SimulationOptimizationModel to run unique experiments based on the specifications given to these three objects. Recall that in an execution of the algorithm, one objective function evaluation requires the execution of one simulation experiment. This way, the number of the SimBasedOptExperiment instances created is going to be equal to the maximum number of evaluations that the algorithm is to execute, while there is only one SimulationBasedProblem object.

![Figure 4 High level UML class diagram of SIMEON framework](image)

As depicted on the figure above, SIMEON is constructed by Java implementations of the three layers previously presented in the system architecture (Figure 3). In the presentation layer, all visualization objects are contained. In the problem processing layer, the simulation and optimization engines with their main object components are grouped in different sub-systems. Finally, the objects from the knowledge layer are not presented here as they are problem specific. We can therefore clearly see that the design of the framework above is a translation of SIMEON system’s architecture.

4.2.2 Design of genetic representation and operators

Based on the high-dimensionality requirement, two types of decision variables have been identified: quantitative and qualitative decision variables. A simulation-based problem/model might contain quantitative decision variables, qualitative decision variables or both types at the same time. To enable SIMEON having the desired generality and yet also having the ability to solve the problems efficiently, different genetic representations and operators are implemented. The central idea here is to enable SIMEON users to select the most efficient representations and operators based on the types of the decision problems that are
encountered. Following are the specifications of all genetic chromosomes implemented in SIMEON.

1. Binary representation. This is particularly designed to solve continuous optimization problems. Hence, mapping from chromosome to solution is done from binary to real values. This implementation allows SIMEON to have an alternative representation next to real representation to solve continuous optimization problem. To perform the search process, single point crossover and bit flip mutation are implemented for this representation.

2. Real representation. Real number representation is best used to solve problems with quantitative decision variables as it uses natural representation of the solution [21]. Furthermore, we also implement specific operators for this representation: the simulated binary crossover [13], and polynomial mutation [11].

3. Integer representation. Integer representation is designed to solve problems that contain decision variables that are qualitative in nature such as policy alternatives, type alternatives, etc [3]. It is noteworthy that in the context of simulation-based optimization, structural alternatives of a model can be represented into integer decision variables which SIMEON is able to provide an optimization service. The search process of this representation is supported by single-point crossover and bit-flip mutation.

4. Integer-real representation. This is a form of solution encoding that makes use integer and real representations simultaneously. This representation is particularly useful to solve simulation-based or mathematical model-based problems that contain both quantitative and qualitative decision variables. To perform crossover and mutation for this representation, two operators are specially implemented: Integer-Real crossover and Integer-Real mutation. These operators perform different genetic operations on different segments of the chromosome depending on the types of the decision variables. The crossover operator combines single point crossover and simulated binary crossover while the mutation operator combines bit flip mutation and polynomial mutation. Figure 5 Illustrates this representation.

4.2.3 Design of the SIMEON User Interface (SIMEON UI)
User interface (UI) of SIMEON is the main constituent of the presentation layer. It plays an important role to take and present messages from and to the user through various language and presentation sub-systems. Therefore, its design is vital for the usability of SIMEON for both expert and non-expert users. The following features are implemented for the UI:

1. SIMEON control panel
Control panel has two main functions:
   a. Allowing the user to configure the algorithm and the problem.
   b. Visualizing the non-dominated solutions for either 2 objective functions (on the 2 objective functions plot) or multiple objective functions (on the spiderchart plot).

On the top-left panel, the user gets to configure the algorithm to use and its parameters. Next, on the bottom-left panel, the simulation-based problem to solve can be selected from a dropdown menu. The run-control conditions can be specified in the same panel. The right panel is used to show the non-dominated solutions for two objective functions.

2. SIMEON modeling interface

The modeling interface provides a separation of concerns between modeling the outcome measures of the simulation model and modeling the optimization problem based on those outcome measures. This allows the user to make custom-made objective and constraint functions depending on the objective of the optimization study, giving the user the flexibility to do experimentations with different optimization models. Furthermore, these models can be saved and loaded to and from a file.
3. Input and output Excel sheets
We use excel sheet as an example implementation of connecting SIMEON to a widely-used external software. With this sheet the users can specify the decision variables and get the result of each execution.

5. SIMEON OPERATIONAL TEST AND ASSESSMENT
In this section, we present two simulation-based problems that we use as the test cases for SIMEON. The first test case comes from the supply chain management. This case only contains continuous decision variables. The second test case is a well-known manufacturing optimization problem, namely scheduling. This test case involves discrete and continuous decision variables.

For both tests, we use a population size of 100; 10,000 evaluations; a 0.9 crossover rate, 1/number of variables mutation rate, and 20 distribution index (for the polynomial mutation and simulated binary crossover) to perform the optimization. Finally, we perform the test cases on a notebook with Intel Core 2 Duo processor, 2.26 GHz, and 3072 MB memory.

5.1 Supply chain optimization problem
The supply chain optimization problem is characterized by the presence of multiple actors/companies and the dynamic interactions that take place during the distribution of products, money and information between those actors. In the simulation model used as a test case, there are two suppliers, one manufacturer, three distributors and four market spots/retailers. Figure 8 illustrates this description.

The supply chain process is started when the four customers (Coca cola, Sisi, Hero and PepsiCo) generate demands that follow certain statistical distribution. These customers are served by two distributors (DistriCan and ShipACan) according to the relationship depicted in figure 8. Various transport modes result in different delivery speeds between those actors. Products are produced by a manufacturer (can company) by using materials supplied by two suppliers (Hoogovens and Akzo). The products are first accumulated in a consolidated central warehouse in Brisbane before they are transported to the distributors. The objectives are to maximize inventory unit fill rate and to minimize the supply chain cost of a distributor (DistriCan) for a 100 days period. The decision variable is the service level policy of this distributor. In the simulation model, service level is a decision variable which influences supply chain cost and inventory unit fill rate through a chain of transitions functions. We only perform one replication for this model. Figure 9 shows the non-dominated solutions of the problem.

We can see that there are two non-dominated solutions that can be selected for a decision. The first solution suggests setting the service level rather high (96%) while the second solution suggests a lower service level (87%). The first solution, however, indicates a good performance on the inventory unit fill rate (93%) with relatively higher cost (28 cost unit) while the second solution gives lower cost of the supply chain (22 cost unit) with relatively lower inventory unit fill rate (89%). Given this information, the decision maker(s), can have the assurance of picking a solution out of the best alternatives possible. Further process of decision making may use additional preferences of the decision maker(s) depending on their specific target or circumstances.

5.2 Flexible Manufacturing Scheduling problem
The scheduling problem dealt here is taken from the domain of flexible manufacturing system (FMS) where the machines inside such a system have the flexibility to change the order of operations executed on the products. In this test case, there are five different products that have to be processed through three different machines (e.g. drilling, painting, and finishing), before eventually being disposed as finished products. Each of those products has a unique routing sequence throughout the machines, and a unique processing time which follows certain statistical distribution. Each of the machines has four different priority rules that can be set to handle the queue of products that enter each of those machines. These are First In First Out (FIFO), Last In First Out (LIFO), and Shortest Processing Time (SPT), and Longest Processing Time (LPT). Figure 10 illustrates the discussed FMS.
The objectives are to maximize the total production of parts and to minimize the cost of the production in a 50 hours span. Next, the constraint is that we only accept solutions that have a total production larger or equal to 60 units. The decision variables of the system design are the priority rules that are to be set for all three machines and the level of processing quality for the manufacturing system. While the calculation of the total product depends on the combination of priority rules (as the controllable initial states) and the transition function within the simulation model, the calculation for the cost depends on the level of processing quality which has a negative correlation with the cost itself. We use 3 replications to smooth out the stochasticity of the model.

To perform the optimization using integer-real encoding, we encode the qualitative variables “priority alternatives” as integer numbers that range from 0 to 3 representing FIFO, LIFO, SPT and LPT respectively. On the other hand, the level of processing quality is encoded as a real number ranging from 5 to 15. Furthermore, 3 replications are used to take into account the stochastic nature of the model. Figure 11 shows the optimal solution of the problem.

Figure 10 Flexible manufacturing system with four work stations

Based on the result presented above, there is only one optimal value for the two objectives (81 products and 25 cost units). In this case, the decision maker does not need to make any trade-off between the two objectives. This is because the two objectives in the simulation model do not have any correlation with one another. Furthermore, two system configurations are equally optimal:

1. Setting the priority rules to FIFO for all the machines and setting the level of processing quality to 5.
2. Setting the priority rules to SPT, LIFO, and LPT for drilling, painting and finishing machines respectively and setting the level of processing quality to 5.

6. CONCLUSIONS AND FUTURE RESEARCH

In this research we have identified the knowledge gap in the integration of simulation and optimization techniques. Furthermore, we also presented the challenges in developing a simulation-based optimization method that fulfills generality, efficiency, high-dimensionality and usability features.

To address those problems, we design and implement SIMEON, a framework designed based on the aforementioned features. The conceptual design of the SIMEON framework has successfully provided the answer for the knowledge gap. We presented the definitions and the structure that make clear the way optimization and simulation techniques can be integrated in a theoretically sound framework.

Next, based on the conceptual design, we presented the system architecture and the detailed design of SIMEON. Two cases are used to test the performance of SIMEON. The results show that the SIMEON framework is able to realize the desired features.

As for the future research, we plan to further develop SIMEON with different evolutionary-based algorithms and perform further testing with real-world and benchmark problems.

7. ACKNOWLEDGMENTS

8. REFERENCE
