SIMULATION OPTIMISATION USING SIMKIT

Serdar Bozoglan
Murat M. Günal
Institute of Naval Science and Engineering
Turkish Naval Academy
Tuzla, Istanbul, Turkey.
zserdarb@hotmail.com , mgunal@dho.edu.tr

ABSTRACT
In this paper, we present early findings of an ongoing research which investigates the use of an open source simulation library, Simkit, to develop and implement simulation optimisation algorithms. Simulation based optimization is an available feature in most COTS simulation software. However, in these software different search strategies are adopted, which does not allow users to compare performance of optimization algorithms. In our tool, we allow users to choose one of the available heuristics, as well as to do complete enumeration, if it is feasible, to search for the optimum response. We used a well-known problem as a test-bed for our tool.

Key Words: Simulation Optimization , Metamodel, SIMKIT, Java.

1. INTRODUCTION

Simulation and optimization have been thought together for many years. Whilst optimisation techniques are used to search for the best by mathematically programming a problem, simulation is used to understand complex systems by modelling. Optimisation techniques are known with their speed, and simulation is known with its flexibility and its capability to handle the complexity. Coupling these two operations research techniques emerged a new research area (Law (2000) and Fu (2002)), which also affected the practice. Nowadays almost all commercial discrete-event simulation software have an optimization module.

The main idea of simulation optimisation (SO) is to use the two techniques, optimisation and simulation, inline. That is the output of a simulation model feeds an optimisation model, and later new solutions are fed to the simulation. New solutions are obtained from initial solution and this new obtained solutions are used as input parameters so this iterative approach goes on until achieving a stopping rule. Simulation model can be thought as a black-box method in the metaheuristic optimisation model. Figure-1 shows this approach. The metaheuristic optimiser uses a set of values, that is the output of a simulation model, and these values are evaluated in the optimiser to suggest better solutions. The optimiser does not find an optimum at once, but rather searches the solution space.

We adopted the general approach in Figure 1 and aimed at developing an optimiser module which can be plugged to a simulation model. The module is only capable of problems where decision variables are integers and a feasible solution can be represented as a vector of integers. In our optimisation module, we will include three heuristic algorithms, genetic, tabu search, and simulated annealing. To get a general view of solution space we also allow users to run experiments with partial or complete enumeration of solution space.

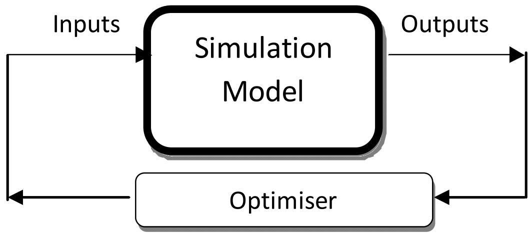

Figure 1 Black-box approach to optimization via simulation

On the simulation model side, we used SIMKIT as our modelling software. Simkit is an event-driven open-source Java package developed by Buss (2002). Using Simkit, the modeller focuses on events that causes state changes in a system. The modeller writes a method for each of the events in system. Simkit is particularly powerful in combat modelling.

In the next section a brief review of the literature is given. Simkit and the optimisation module are explained in more detail in section 3. In section 4, an example which demonstrates the use of the module with Simkit is presented.

In this research, we aimed at investigating the benefits of having multiple heuristics algorithms in a simulation optimization package. The users of this package can compare different algorithms to solve optimization problems, and therefore can evaluate which algorithm suits to the problem on hand. As best of our knowledge, the simulation software in the market, and in the academia, use one optimisation algorithm and does not allow users to try multiple algorithms. We aimed at filling this gap in the literature and contribute to the discussion.

2. SIMULATION OPTIMISATION: A REVIEW

The literature in this area is rich and it is not surprising that there are already reviews and classifications of the studies conducted. Sabuncuoglu and Tekin (2004) is one of such reviews where the methods and approaches in SO are classified and examined in detail. Our intention here is not to give a complete re-review of the literature, but to let the readers refresh their knowledge.

As in a classical optimisation problem, there are one or more objective functions and some constraints in a simulation-based optimization problem. The objective function consists of decision variables, which are input parameters of a simulation model. These input parameters are known as “factors” and they are transformed to output performance measures called “responses”. Structural assumptions can also be seen as factors in a simulation model. Factors can be classified as qualitative or quantitative; Quantitative factors have numerical values like number of machines in a system or reorder point in an inventory system. Whereas qualitative factors contains structural assumptions like queue priority or routing policy in a system. In addition to this classification, it is possible to categorize factors as controllable or uncontrollable (Law, 2004).

Experimental design used in SO aims at discovering which factors have the most effective influence on the response and finding the right combination of factors which maximizes or minimizes the response (Law, 2004). It is important to narrow down the search space, by smartly designing the experiments, since each experiment requires computing time. More experiments mean more model run and more time. The efficiency is therefore crucial in SO. Lou et al (2000) grouped the efforts to increase the efficiency into three; modelling, analysis, and execution.

As mentioned earlier there are many methods and techniques in SO which are available in modellers’ toolkit. We simply name some of these methods here. If the optimization problem has a single objective function then the methods listed below are most commonly used;

- Gradient Based Search Methods
- Stochastic Approximation Methods
- Sample Path Optimization
- Response Surface Methods
- Heuristic Search Methods

In the case of multi-criteria optimization, special methods such as the ones listed below are used;

- Variation of Goal Programming
- Multi-Attribute Value Function Method
- Using one of the responses as the primary responses that depend on the achievement of other objective functions.

Our general review also includes pros and cons in SO. The advantages of simulation in optimization can be grouped under three headings (Azadivar, 1999):

- **Complexity**: Modelling complex systems for solving problems is easer in simulation than in optimization. Simulation is known as a technique with its capability of handling the complexity in systems. However it is difficult to integrate detailed features of systems in optimization problems.
- **Stochasticity**: Factors generally have stochastic nature. In case of stochastic systems, the variance of the response can be controlled by using some output analysis techniques. Whereas in optimisation it is hard to include stochastic elements in a system.
- **Flexibility**: Simulation is also known with its flexibility to address different situations. It is easy to model real life problems. This feature of simulation is also applicable in SO.

Although there are advantages of using simulation for optimisation problems, it brings some issues. For instance, in addition to stochastic nature of factors, complex and highly non-linear optimization problems can be too complex for simulation analysis. The disadvantages of using simulation for optimisation are:

- There is no any analytical expression of the objective functions or the constraints. In this case it is not likely to differentiate or exact calculate the local gradients.
- Computer simulation runs take much more time than evaluating analytical functions. So efficiency of the optimization algorithm becomes vital.
- Optimization and simulation use different computer languages so analyst may have a challenge with that.

3. SIMKIT AND THE OPTIMASATION MODULE

Simkit is a discrete event simulation library written in Java (Buss, 2002) which implements event scheduling paradigm. The modeller can create a model of an event graph using Simkit. An event graph has two basic elements; events (nodes) and scheduling edges (edges). It works as follows: when an event occurs, after a delay and if a condition is met, another event is scheduled. The conversion between an event graph and Simkit model is simple. The modeller writes a method for each event and inside this method next event is scheduled. Then the simulation executive runs the event list.

```
While optimisation not finished
   Create one solution
   For ReplicationNum=1 to NumRuns
      Stop at (Run Length)
      Reset variables
      Run Simulation
      Collect one Run Stats.
   Collect one Experiment Stats.
   Record the response of one solution
```

Figure 2 Pseudo code for SO module.

The pseudo code given in figure 2 is the implementation of the simple representation in Figure 1; that is running the model with the input values suggested by the optimiser, based on the response values produced by the model. Since there are stochastic elements, we need to run a model multiple times and collect statistics to estimate mean response values (the inner loop). The outer loop manages the search for the optimum. It can be a complete search in the search space or a heuristic search. The decision of stopping the search depends on the search algorithm. It can depend on number of iterations, real time or near value. In the worst case it can be complete enumeration.

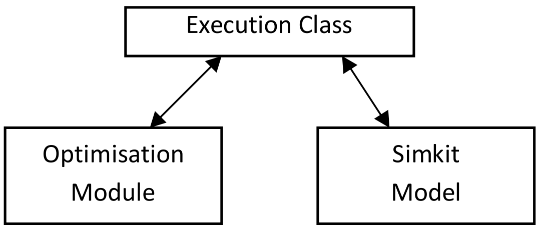

Figure 3 The link between optimisation module and a model.

The Simkit model and the optimisation module are separated from each other. They communicate via an execution class which only keeps a main method. The optimisation module, on the other hand, evaluates response values received from the execution class and generates new solutions which then transfer to the model. A solution means a new input set of values. This separation allows the optimisation module to be runnable in different modelling software.

4. A SIMPLE MODEL AS A TEST BED FOR PARTIAL ENUMARATION

4.1. Introducing the problem

To test our methodology, we chose a simple simulation optimisation problem from the literature. Law and McComas (2000) have first introduced this problem which, later, is included in Law (2007, p.663). In this problem, there are four serially connected workstations and in each workstation there is finite number of machines. The number of machines in each workstation is one of the two decision variables. Machines process blank parts and pass the parts to the buffers, which have finite capacities, between completed workstation and the next workstation. Hence there are three buffers. The buffer capacities between workstations are the other decision variables. In this example it is assumed that there is infinite supply of blank parts and the first workstation can never be idle. However, one complication is that workstations can be blocked when all of the buffer slots of the completed workstation are full, in which case the workstation cease processing.

In this optimisation problem, the constraints are related to the number of machines in each workstation, which can be up to 3, and the number of buffer slots, which can be up to 10. The later is associated with the space capacity. The objective is to maximize the profit which is given as a function of throughput (number of parts produced), number of machines, and number of buffer slots.

4.2. The Simkit Model

The starting point to develop a model using Simkit is to draw an event graph of the system to be modelled. Note that this is also one way of representing a conceptual model of a system. The Simkit model includes arrival of parts events, start and end of machine events. Number of machines and number of buffers are countable resources. Arrival process in this particular problem is tricky since an arrival to a machine occurs if the machine is idle and if the succeeding buffer is free.

4.3. Experimentation

In this type of problems, where decision variables can have discrete values, enumeration is feasible. Initial values can be assigned to decision variables and in each iteration of the outer loop in fig. 2, values can be incremented or decremented within the constraints of these variables. If every possible value is tried then a complete enumeration is achieved. This will ensure that each solution in the solution space is searched and an objective function value is calculated for every possible solution. Although this approach sounds logically good, it may not be practically appropriate since the number of solutions in the solution space can be too many and hence complete enumeration can take too much computing time. For this reason optimisation techniques look at smarter ways of searching by restricting the search space and by directing the search to the areas where the optimum is more likely to be.

Number of solutions in the solution space is a function of the number of decision variables and their ranges. The range for a decision variable is meant to be the constraints in an optimisation problem. For example in our problem we restricted the number of machines and the buffer sizes. These limitations significantly restrict the search space. Four workstations and up to three machines in each mean that we have 3^4 alternative configurations, and likewise, three buffers and up to 10 slots in each mean that we have 10^3 alternative configurations. In total we end-up with 81,000 combinations of seven decision variables.

We experimented with Simkit model for 10,000 solutions which were chosen randomly. As discussed before complete enumeration is possible but not practical. Our partial enumeration means that almost 1/8 of the whole space is scanned. Results of this experimentation are shown in fig. 4.

4.4. Performance

We conducted our experiments on a laptop with 2 Gb Ram and 2.2 Ghz, Intel Core 2 Duo processor. We examined a number of cases where number of solutions is changed. In these experiments we run the model 5 times.

As seen in table 1, the model performance is directly correlated, in fact linearly, with the number of solutions examined. 10000 solutions take 5400 seconds to complete.

Table 1 Model performance in different number of solutions.

Number of solutions	Time
1	0.6
10	5.5
100	53.0
1000	520.0
10000	5400.0

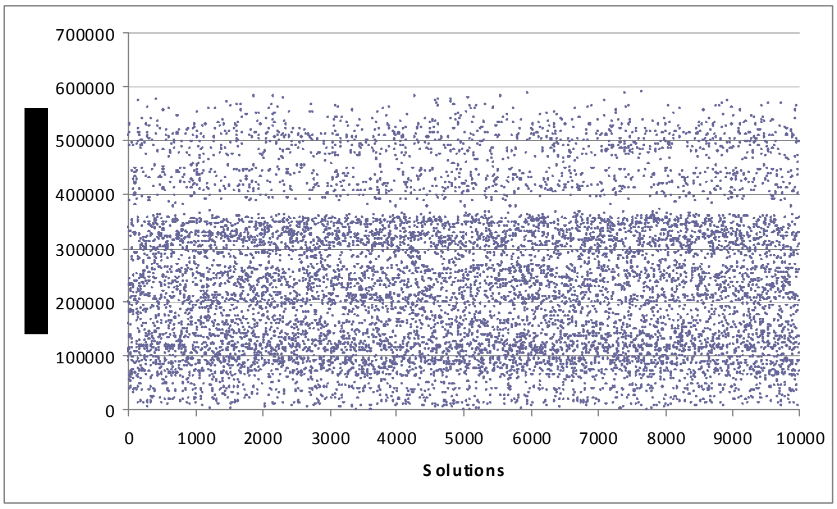

Figure 4 Output of 10000 runs.

5. FUTURE RESEARCH

In this paper, we briefly reviewed the techniques for simulation optimization and presented preliminary outcomes of our study.

In the design of our package, we foresee adding some well-known heuristic algorithms, such as Genetic Algorithm (GA), Tabu Search (TS), and Simulated Annealing (SA). We aim to have a heuristic optimisation package which can be plugged to a simulation model. Although the problem types that this package can be used for is limited (integer decision variables) it is believed to be useful for comparing heuristic algorithms in simulation optimisation problems.

ACKNOWLEDGEMENTS

This paper is written as part of the first author's master's thesis study.

REFERENCES

April Jay, Glover Fred, Kelly P.James, Laguna Manuel (2010), The Exploding Domain of Simulation Optmization, www.opttek.com [as accessed on Jan.2010]

Azdivar Farhad (1999), Simulation Optimization Methodologies, Optimization, Proceedings of the 1999 Winter Simulation Conference

Buss, A.(2002), "Component based simulation modeling with Simkit," Simulation Conference, 2002. Proceedings of the Winter , vol.1, no., pp. 243-249 vol.1, 8-11 Dec. 2002.

Deng Geng (2007), Simulation-Based Optimization, PhD thesis, University of Wisconsin- Madison

Fu Michael (2002), Optimization for Simulation Theory vs. Practice, INFORMS Journal on Computing/Vol. 14, No. 3, Summer 2002

Law Avreill, McComas G. Michael (2000), Simulation-Based Optimization, Proceedings of the 2000 Winter Simulation Conference

Law A.M. (2007), Simulation Modeling and Analysis, Fourth Edition Publisher: Location

Lou Yuh-Chuyn, Chen Chun-Hung, Yucesan Enver, Lee Insup, (2000), Distributed Web-Based Simulation Optimization Proceedings of the 2000 Winter Simulation Conference

Sabuncuoglu İ., E. Tekin. (2004) Simulation Optimization: A Comprehensive Review on Theory and Applications, IIE Transactions, Vol: 36, pp: 1067-1081, 2004.

AUTHOR BIOGRAPHIES

SERDAR BOZOGLAN is an MSc student in the Naval Operational Research masters programme in the Institute of Naval Science and Engineering. He holds a BSc in Industrial Engineering degree where he received in 2002 from the Turkish Naval Academy.

MURAT GUNAL is a lecturer in the Turkish Naval Academy and also teaches in the Institute of Naval Science and Engineering. He completed his PhD and MSc studies in Lancaster University, UK, in 2008 and 2000 respectively. His PhD thesis' title is "Simulation Modelling for Performance Measurement in Hospitals". He did research and worked in simulation area many years. His current research interest is simulation optimisation.