

# SOLVING LOGISTIC PROBLEMS THROUGH SIMULATION AND EVOLUTION

Peter Köchel

Chemnitz University of Technology, Faculty of Computer Science, D-09107 Chemnitz  
[pko@informatik.tu-chemnitz.de](mailto:pko@informatik.tu-chemnitz.de)

## Abstract

Simulation Optimisation (SO) is suited for solving such optimisation problems, which cannot be solved by conventional approaches. The paper briefly introduces to the main ideas of SO and as a concrete realisation discusses the combination of simulation and Genetic Algorithms. Finally, the proposed approach is applied to solve a complex multi-location inventory problem.

**Keywords:** optimisation, simulation, Genetic Algorithms, multi-echelon inventory model

## 1. Introduction

From the historical viewpoint the necessity to solve logistic problems was one of the origins of Operations Research. Today's systems, which had to be designed and to controlled in an optimal way, are so complex that traditional analytical approaches alone lead to non-sufficient solutions only. Therefore we suggest combining methods from Operations Research with those from Computer Science. One result of such a combination is *simulation optimisation*, whereby a simulator for the real system to be investigated will be coupled with optimisation methods suitable for the given problem.

Simulation optimisation is a relatively new research field. The supply of simulation packages with optimisation tools has started a few years ago and is today one of the current trends [BANK00]. Some examples are given in [FUMC01]. As optimisation methods are quoted Genetic Algorithms (GAs), Scatter Search, Tabu Search, Neural Networks and Simulated Annealing. Besides these heuristics exist in the literature several statistical procedures and stochastic optimisation methods, which realise a gradient search known from deterministic optimisation. A survey of these methods contains [FUMC94], whereas [PFLU96] gives the strong mathematical foundation. Though the theory on simulation optimisation has reached a satisfying level the application is exceedingly unsatisfactory. Up to now basically no problem with practical relevant size is solved by simulation optimisation. With respect to its application especially to inventory and logistic problems from the existing literature is formed the following picture. Very simple inventory models (one location, linear cost functions, one product) serve as demonstration examples for stochastic optimisation methods [FUMC94]. Most of publications concentrate upon single aspects of simulation optimisation without considering the problem as a unity [BOES01]. Furthermore, little or no support for logistic and inventory processes in simulation software meets no or insufficient know-how about simulation and optimisation in corresponding companies [MANI98], [BOES01]. On the other hand it is shown for multi-location inventory systems in [ARNO96] and [KOEC03b] and for logistic problems in [KOEC03a] that the combination of simulation and GAs is practicable on principle.

The aim of the present paper is to give a brief introduction into the basic principle of simulation optimisation and to show the applicability of that approach. Chapter 2 contains these basics. The emphasis lies on GAs as optimisation tool. We discuss not only why we use GAs but also several advantages and disadvantages of our approach. As an example for the broad range of application we consider in Chapter 3 how to control in an optimal way the flow of product in a multi-echelon model. Finally, in Chapter 4 we give a summary and an outlook for further work.

## 2. Optimisation through simulation and evolution

Let an optimisation problem be defined by the pair  $(f, D)$ , where  $f$  denotes the criterion function and  $D$  the set of admissible decisions  $d$ . Most optimisation methods are based on the assumption that  $f$  is given in an analytical tractable form and that its values can be calculated exactly. For complex systems such an assumption does not hold no more. Values of function  $f$  can be calculated only as the output of a simulation experiment. In the case of stochastic systems one will get even estimations only. However, calculation or estimation by a simulation experiment does not solve an optimisation problem. For that one has to combine a corresponding simulator for the system to be optimised with an appropriate optimisation tool. This approach is called *simulation optimisation* (SO). The principle of SO is outlined in Figure 1. The scheme in Figure 1 illustrates that optimisation and simulation alternate in such a way that for each decision suggested by the optimiser the simulator collects information on the performance of that decision. Optimisation and simulation are pushing each other until an acceptable decision is found. From the scheme in Figure 1 follows the main question: How to realise the optimiser? General, if the interface

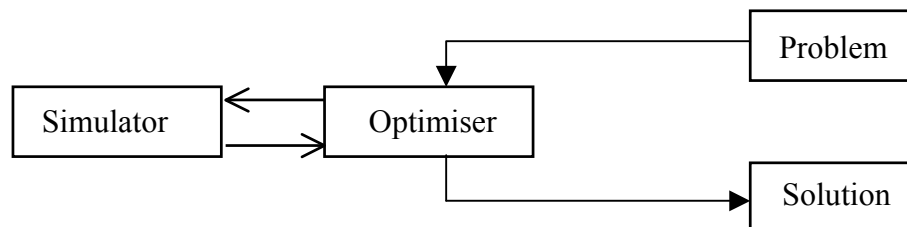


Figure 1. The principle of simulation optimisation

between the simulator and the optimiser is well defined then with a given simulator an arbitrary optimisation method can be coupled. In all our implementations ([ARNO96], [KOE02], [KOE03a], [KOE03b]) the optimiser consists of four parts (see Figure 2).

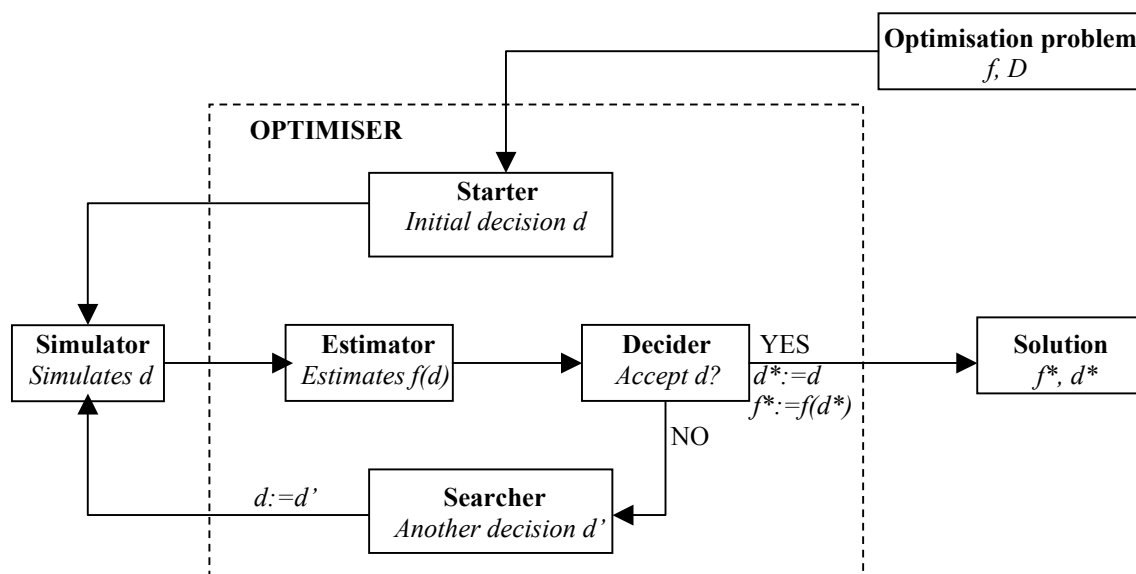


Figure 2. The scheme of the simulation optimisation

The **Starter** chooses an initial decision, which will be given to the simulator. As a result of a simulation experiment with the system configuration, corresponding to the chosen decision, we get a concrete sample of system histories. This information the **Estimator** transforms into an estimate for  $f(d)$ , the performance of the decision  $d$ . On the basis of that estimate, and possibly further available information, the **Decider** else accepts the just investigated

decision as (approximately) optimal and the search process is stopped or the *Searcher* gets the order to find a new and possibly better decision. Thus the search process for an optimal decision will be realised by repeated processing of four stages – proposal of a solution, generation of relevant for the problem data by a simulation experiment, performance analysis on the basis of these data, decision to accept the proposed solution or to continue the search process. That cycle will be passed through until a stopping criterion will be fulfilled. We remark that once started the search process runs automatically without interaction of the user. After leaving the cycle the best of all considered solutions will be returned. The output of the whole process can be broader, e.g., it can be returned the second best solution and so on.

From Figure 2 it will be clear that our approach allows a sequential procedure, where in dependence on the requirements and the ongoing optimisation process further data and information can be included *during* the search process. Thus the proposed approach will be very flexible in such a way, that it allows an efficient distribution of finite resources, that it can use already obtained information, and that it can surmount local optima. Furthermore, Figure 2 points to the main problems that must be solved applying SO. The first problem is how to find an initial decision. In many cases a solution is obvious, e.g., for an inventory problem an initial decision may be to have zero inventories. In cases where set  $D$  is defined by a multitude of complex constraints it may be necessary to develop a special algorithm for defining the initial decision. The second problem is how to estimate the performance of a just simulated decision. Usually will be computed sample averages with corresponding confidence intervals. The next problem is how to decide for stopping or for continuation of the search process. The simplest solution is to define a stopping criterion like a finite number of different investigated decisions or finite computing time or something like this. Finally, the searcher has to solve the most important problem – the definition of another decision in case that the search process will be continued. The solution of that problem is more or less the core of the optimiser in SO. Implementing different algorithms at this place in an essential way influences the quality of the whole optimisation process above all the performance of the finally accepted decision and the computing time required for finding that decision.

For a concrete realisation of the described elements of the optimiser we prefer GAs. In all our investigated problems we applied a GA that was taken from the evolutionary optimisation tool *Laboratory for Evolutionary Optimisation (LEO)*, developed at Chemnitz University of Technology (see [NIEL99]). To represent a decision within the optimisation process, LEO uses a number of chromosomes. Those chromosomes in general are vectors of bits (classic GA) or reals (real-coded GA) or integers or permutations (for combinatorial optimisation problems). The optimisation process in LEO contains five steps. The Starter from Figure 2 is represented by step 0. In that step the initial population of a given number of decisions' representations (chromosomes) is generated. Assigning random values to the components of a chromosome can do this. In step 1, for each of the chromosome types of the representation one corresponding genetic operator is chosen. For instance, the one-point crossover takes two parent-chromosomes  $x' = (x_1', \dots, x_M')$  and  $x'' = (x_1'', \dots, x_M'')$ , randomly chooses an integer  $t$ ,  $1 \leq t \leq M$ , and produces the two child-chromosomes  $y' = (x_1', \dots, x_{t-1}', x_t'', \dots, x_M')$  and  $y'' = (x_1'', \dots, x_{t-1}'', x_t', \dots, x_M'')$ . Apart from the classical crossover and mutation there exist special operators that can be applied to number chromosomes. Altogether nine operators for number chromosomes and eight for bit and permutation chromosomes are implemented in LEO. The application of such an operator takes some parent-chromosomes and produces some child-chromosomes. In step 2, the required number of parent-decisions are selected using the selection methods. They are all implemented both as better-selection to prefer decisions with better objective value and as

worse–selection to prefer decisions with worse objective value. In step 3, the child–decisions are created using the operators chosen in step 1. In step 4, the child–decisions are taken into the population by worse–selecting and deleting other decisions from the population. Thus the steps 1 to 4 realise the Searcher - they choose new decisions. To avoid premature convergence, no duplicate decisions are inserted into the population. The concept additionally includes elitism, i.e. the best decision found so far cannot be lost again. We have to remark that the steps 2 and 4 are based on the estimated criterion value for the chosen decisions or chromosomes. The corresponding simulator will do this. The Decider from Figure 2 is realised simply through the definition of a stopping criterion for the GA. Such a criterion can be a finite number of investigated decisions. The search can be stopped also if e.g. among the last 200 considered decisions could not be find a better one than found up to that moment. In LEO, also multiple isolated populations are possible with temporary migration of decisions from one population to the next. LEO also builds a family tree up to a certain depth of all generated decisions to record, which of the genetic operators were used to create that decision and which of selection methods were used to choose its parents before. This family tree can be evaluated to find out, which of the genetic operators and selection methods performed good so far on the actual optimisation problem by producing better decisions out of worse ones. The aim is to apply them more often to increase the efficiency of the optimisation process.

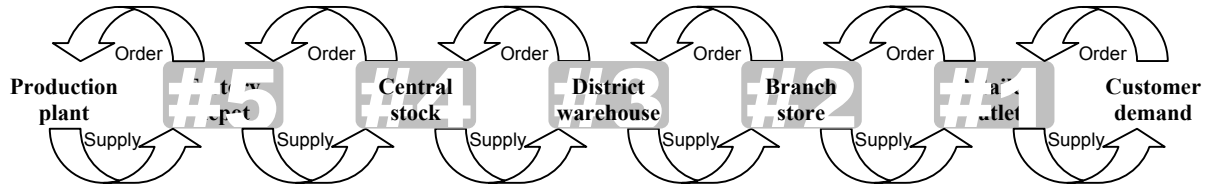
The suggestion to use a GA is based on both our experiences with and the main advantages of GAs. Let us briefly discuss these advantages.

1. GAs have in principle an unlimited application area, i.e., GAs can solve optimisation problems where other methods will fail (not well-structured set  $D$  of admissible decisions; a decision can include as well real and discrete components as structural or qualitative components; criterion function  $f$  must not be given in an analytical form).
2. GAs have such nice properties like robustness with respect to starting points, excellent handling of the random output of simulation experiments, and a high probability for locating global optima even if there exist many local optima.
3. GAs can be designed, tested, and tuned independently of the application domain.
4. GAs need only a small amount of input information. Thus the interface to simulators can be kept very simple.
5. GAs possess an inherent parallelism.

These advantages are faced with three main disadvantages, which partially hold also for other solution methods. To apply a GA we have to parameterise the optimisation problem in such a way that a decision can be represented as a vector. However, in most cases this should be possible. However, the requirements for resources strongly increase with increasing dimension of the parameter vector. Finally, since a GA realises a stochastic search we will get in finite time with positive probability only a sub-optimal solution. For more information on GAs see for instance [MIET99].

### **3. An Example: The Multi-echelon model**

In this chapter we apply the proposed approach to a 5-echelon inventory system investigated in [KOEC03]. In a multi-echelon inventory system (MEIS) the flow of products goes from the first echelon or stage, the producer, through several intermediate stages to the final stage, the retailer, which has to serve a random demand (see Figure 3). The optimisation problem is to define such an order policy for each echelon that the expected total cost per time unit will be minimised. [CLAR60] is the first paper on MEIS. Under some assumptions like linear cost functions and discrete time, measured in periods, an exact solution could be found for a serial system. Dynamic programming in general and Markovian decision theory in particular were the main investigation methods for periodic



**Figure 3. Layout of the exemplary echelon inventory system**

systems. Later on systems with continuous time are investigated. With the 1990<sup>th</sup> the research on analytically solvable MEIS came to a defined end. A state-of-art review on continuous time and discrete time systems give [AXSA93] respectively [FEDE93]. At the same time two new aspects came up. The first is an ecological aspect. In addition to the goods flow towards the retailer it becomes more and more important to consider the backflow of used materials (packaging materials, containers and so on). The actuality of this *reverse logistics* continues to increase. The other aspect is the influence of computer science and information technology. The information technology allows both to share information on the whole system and to deliver full information to each echelon. Computer science developed such methods like Artificial Intelligence and several heuristics and meta-heuristics. Thus it was possible to consider more realistic systems and to overcome the constraints of analytically based approaches. Our small example should demonstrate this.

We assume a single-item model with Poisson demand with a rate of 0.02 per minute. If at the retailer outlet is no product then up to 10 demand units can be queued in a waiting line for waiting cost of 4 per minute and backlogged demand unit. If an arriving demand unit can not find a free waiting place it is rejected for rejection cost of 2 000 per rejected demand unit. Further data for all locations are given in Table 1, where C, N and U stand for a constant, a normal and a uniform distribution, respectively. To apply the SO-approach we

Echelon stage	#5	#4	#3	#2	#1
Holding cost per item and minute	1	1	1	1	1
Packaging time per item in minutes	N (10;5)	U [30;90]	N (40;10)	C = 20	U [5;15]
Transportation time to stage in minutes	C = 10	C = 50	C = 250	C = 100	C = 60

**Table 1. Further cost and time data for the exemplary echelon inventory system**

have to parameterise the optimisation problem. For this we will investigate only continuous-review, order-point, order-quantity strategies  $(s, Q)$ , where  $s$  denotes the order point and  $Q$  the order quantity. Applying such a policy at a given echelon means that the echelon orders from the upstream echelon quantity  $Q$  if the local inventory position has reached or is below  $s$ . Assuming that all echelons have full information on the status of the whole system, especially on the waiting demand, we compute the local inventory position as the sum of the echelon stock and the outstanding upstream orders, where the echelon stock of some echelon is the number of items in the system that are at, or have passed through, that echelon but have as yet not been specifically committed to outside customers. Finally, we assume fixed order cost only – 1, 25, 200, 150, and 20 for echelon 5, 4, 3, 2, and 1 respectively. For the described problem a decision vector (chromosome) consists of 10 components, i. e.,  $x = (s_1, Q_1, \dots, s_5, Q_5)$ . For each decision to be tested, three simulation runs estimated the total costs per year. Each run consists of a transition phase and of 700 days of simulated real time. During that time approximately 20 000 demand events were simulated. All data then were transformed to a one-year time period. The optimisation process by the Genetic Algorithm was stopped after considering 5 000 solutions. In Table 2

Echelon stage	#5	#4	#3	#2	#1
<b>Order point <math>0 \leq s \leq 20</math></b>	<b>5</b>	<b>16</b>	<b>18</b>	<b>11</b>	<b>10</b>
<b>Order quantity <math>1 \leq Q \leq 100</math></b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>12</b>

<b>Total costs per year</b>	<b>26 825 367.642</b>
Waiting costs	10 335 030.711
Rejection costs	3 800 788.800
Transportation costs	1 394 365.291
Holding costs	11 295 182.839
Served customer orders	8 470.920
Rejected customer orders	1 900.394

Configuration found	4 736-th of 5 000
---------------------	-------------------

**Table 2. Optimisation results for the multi-echelon model**

we can see the results. Together with the best-found parameter values Table 2 contains also the estimated values for different cost parts. This is an additional advantage of simulation – we can investigate the changes of various performance measures inside of a single simulation experiment.

#### 4. Conclusion

Based on a brief introduction to Simulation Optimisation we have demonstrated through a simple example from inventory theory how to combine simulation and Genetic algorithms to solve complex control and optimisation problems. Theoretically such a combination will solve arbitrary problems, which admissible solutions can be described by a vector of parameters. However, with increasing dimension of these vectors our approach needs more and more resources for finding a satisfying solution. Consequently the solution process through SO should be substantially refined. This can be done in different ways:

1. With parallel solution processes that will run on distributed hardware the computing time can be decreased. We can develop a parallel simulator and/or a parallel optimiser.
2. GAs can be coupled with other methods to a *hybrid* optimiser. Such an optimiser possesses some intelligence, which enables him to adapt to different problems.
3. If parts of an investigated system can be numerical analysed an appropriate combination of analytical and simulation models will lead to considerable reductions of simulation time.
4. The overall simulation time to estimate the performance of considered decisions can be decreased for instance if at the beginning of the solution process we work with rough estimations, and we increase the exactness of estimations toward the end of the process.

These points are more related to modelling and computer science aspects. Further improvements can be reached by a sophisticated problem analysis. Thus often a reduction of set  $D$  of all admissible decisions is possible. This can be done e.g. through the derivation of additional constraints from the practical background of the problem or through the admission of simple-structured decisions only. Such decisions usually can be characterised by a few parameters, which will accelerate the solution process.

To summarise we can state that the application of SO for solving real problems is only at the beginning. Further work is necessary to improve that approach, especially by exploitation of the potentialities of computer science and the adaptability to concrete problems.

## References

- [ARNO96] Arnold, J./Köchel, P.: Evolutionary Optimization of a Multi-location Inventory Model with Lateral Transshipments. *9<sup>th</sup> Intern. Working Seminar on Production Economics, Igls, 1996, Pre-Prints*, v. 2, 401-412.
- [AXSÄ93] Axsäter, S.: *Continuous Review Policies for Multi-Level Inventory Systems with Stochastic Demand*. Ch. 4 in [GRAV93]
- [BANK00] Banks, J.: The Future of Simulation. *Proceedings of the European Simulation Multi-Conference, Ghent, Belgium, May 23-26*
- [BOES01] Boesel, J./Bowden, R./Glover, F./Kelly, J./Westwig, E.: Future of Simulation Optimization. In [MEDE01], 1466-1469
- [CLAR60] Clark, A.J./Scarf, H.: *Optimal policies for a multi-echelon inventory problem*. *Management Sci.*, v.6 (1960), 475-490
- [FEDE93] Federgruen, A.: *Centralized Planning Models for Multi-Echelon Inventory Systems under Uncertainty*. Ch. 3 in [GRAV93]
- [FUMC01] Fu, M.C.: Simulation Optimization. In [MEDE01], 53-61
- [FUMC94] Fu, M.C.: Optimization using simulation: a review. *Operations Research*, v. 42 (1994), 199-248
- [GRAV93] Graves, S.C./Rinnooy Kan, A./Zipkin, P.: *Logistics of Production and Inventories. Handbooks in OR & MS*. Vol. 4, 1993 Elsevier Sci. Publishers B.V.
- [KOEC02] Köchel, P./Nieländer, U.: Kanban optimization by simulation and evolution. *Production Planning & Control*, v.13 (2002), 725-734
- [KOEC03a] Köchel, P./Kunze, S./Nieländer, U.: Optimal control of a distributed service system with moving resources: Application to the fleet sizing and allocation problem. *Int. J. Production Economics*, v. 81-82 (2003), 443-459
- [KOEC03b] Köchel, P./Nieländer, U.: Simulation-based optimisation of multi-echelon inventory systems. Forthcoming in *Int. J. Production Economics*
- [MANI98] Manivannan, M.S.: Simulation of logistics and transportation systems. In *Banks, J. (ed.): Handbook of Simulation*. John Wiley & Sons, 1998, 571-604
- [MEDE01] Medeiros, D.J./Peters, B.A./Rohrer, M. W./Smith, J.S. (eds.): Proceedings of the 33<sup>rd</sup> Winter Simulation Conference. *Washington D.C.: IEEE Computer Society*, 2001
- [MIET99] Miettinen, K./Neittaanmakt, P./Periaux, J.: Evolutionary Algorithms in Engineering and Computer Science. *John Wiley and Sons*, Chichester 1999
- [NIEL99] Nieländer, U. (1999). Simulation Optimisation of Kanban Systems using a non-standard Genetic Algorithm. In *4<sup>th</sup> ISIR Research Summer School on Inventory Modeling, Exeter, UK*.
- [PFLU96] Pflug, G.C.: Optimization of Stochastic Models. *Kluwer Academic Publishers*, Boston 1996