Transport and Telecommunication, 2018, volume 19, no. 3, 233–243
Transport and Telecommunication Institute, Lomonosova 1, Riga, LV-1019, Latvia
DOI 10.2478/ttj-2018-0020

THE COMBINATION OF DISCRETE-EVENT SIMULATION AND GENETIC ALGORITHM FOR SOLVING THE STOCHASTIC MULTI-PRODUCT INVENTORY OPTIMIZATION PROBLEM

Ilya Jackson[1], Jurijs Tolujevs[2], Tobias Reggelin[3]

[1,2] Transport and Telecommunication Institute (TTI)
Lomonosova iela 1, Riga, Latvia
[3] Otto-von-Guericke University, Institute of Logistics and Material Handling Systems
Universitätsplatz 2, Magdeburg, Germany
[1] jcksnl93@gmail.com
[2] tolujevs.j@tsi.lv
[3] tobias.reggeling@ovgu.de

The paper describes an eventual combination of discrete-event simulation and genetic algorithm to define the optimal inventory policy in stochastic multi-product inventory systems. The discrete-event model under consideration corresponds to the just-in-time inventory control system with a flexible reorder point. The system operates under stochastic demand and replenishment lead time. The utilized genetic algorithm is distinguished for a non-binary chromosome encoding, uniform crossover and two mutation operators. The paper contains a detailed description of the optimization technique and the numerical example of six-product inventory model. The proposed approach contributes to the field of industrial engineering by providing a simple, but still efficient way to compute nearly-optimal inventory parameters with regard to risk and reliability policy. Besides, the method may be applied in automated ordering systems.

Keywords: stochastic inventory optimization, simulation-based optimization, simheuristics, smart solutions, non-binary chromosome encoding

1. Introduction

Modern markets are extremely competitive. Businesses are facing unceasingly growing pressure on both prices and quality. Besides that, the company is required to swiftly respond to stochastic market conditions. Incorrect inventory policy leads not only to corporate losses, but also to overproduction, which is extremely harmful for an industry as a whole. In this regard, traditional “binge-and-purge” inventory policy is not appropriate anymore. Each penny lost in variable costs inevitably entails huge additional expenses along with reduced efficiency. Market competitiveness is simply not compatible with over-purchasing and throwing away an extra product. Reports of the International Data Corporation (Brandel, 2009) undeniably confirm this, namely companies that had utilized smart solutions for inventory optimization, managed to reduce inventory levels by up to 25 %. This fact emphasizes that the inventory optimization is, undoubtedly, a tremendously important task, the execution of which will allow a company to cut broad spectrum of operating costs and increase net profit as the result.

The real-world inventory optimization is commonly characterized by the necessity for nearly-optimal solutions in feasible computing times. That is why, the metaheuristics in general and genetic algorithms in particular are used so widely to define an optimal inventory policy. The world is full of uncertainty, which frequently makes classical deterministic approaches unsuitable due to excessive simplicity. On the other hand, metaheuristics provides a gargantuan arsenal of random search methods and parallelization paradigms. This paper proposes a simulation-driven approach to solve the stochastic inventory optimization problem. Unlike in traditional models, values of the reorder level, safety-stock and the reorder quantity are approached iteratively. The research goal is to utilize a simulation instead of an objective function in traditional form and apply the genetic algorithm to find such simulation adjustments that lead to the optimal output. The proposed combined approach provides a modeller with a tool to deal with real-world stochasticity in unconstrained way and assess alternative candidate-solutions by risk and reliability analysis.

2. The Essence of the Method

As it is mentioned in the recent research (Juan *et al.*, 2015), real-life stochastic combinatorial optimization problems may be reformulated as a simulation in a natural way. Thus, the hybridization of metaheuristics and simulation techniques promises to be an efficient solution of stochastic inventory optimization and inventory control problems. First and foremost, the combination of simulation and metaheuristics is focused on efficiency taking into account stochastic components that may be contained either in the objective function or in the constraints. Such approaches are conventionally called simulation–based optimization (Subramanian *et al*., 2000) or "simheuristics" in recent articles (Juan *et al.*, 2014). In general, the proposed method is applied to solve the stochastic optimization problems of the form Eq. (1) in subject to Eqs. (2, 3, 4) including classical (r, Q) models (Zvirgzdiņa and Tolujew, 2016).

$$\min f(a) = E[TC(a)], \tag{1}$$

$$P(r_i(a) \leq \theta_i) \leq k_i;\ \forall i = 1, 2, 3, \dots, n, \tag{2}$$

$$l_j(a) \leq c_j;\ \forall j = 1, 2, 3, \dots, m, \tag{3}$$

where $\mathbb{A}$ represents a discrete space of feasible solutions $a \in \mathbb{A}$ for the inventory optimization problem. *TC(a)* stands for a stochastic total cost function and *E[TC(a)]* is an expected value of *TC(a)* or another probabilistic measure associated with the cost function (e.g., probability of an undesirable events to occur). Eq. (2) stands for probabilistic constraints related to the problem. For example, the probability that the reorder point r_i happens to be smaller than the emerging demand during the replenishment θ_i must be smaller than k_i. Eq. (3) is resource constraints required to be satisfied. Namely, resources (e.g., inventory capacity) utilized by a solution $l_j(a)$ must not exceed a threshold c_j (e.g., available inventory capacity).

The method aims to utilize a simulation instead of an objective function in traditional form and apply the genetic algorithm to find such simulation adjustments that would lead to the optimal output (Fig. 1). In the proposed method, the iterative searching process of the genetic algorithm has to assess the quality of feasible individual solutions, highlighting the promising ones. The process continues until the search time runs out. Immediately after this, a decision maker selects a final solution among promising with regard to a preferable risk policy. According to Pidd (1998), the simulation provides a natural way to introduce randomness of stochastic process. Furthermore, real-world stochasticity may be modelled throughout the best-fit probability distribution. The distribution may be either theoretical or empirical, without the need to be approximated to normal or exponential.

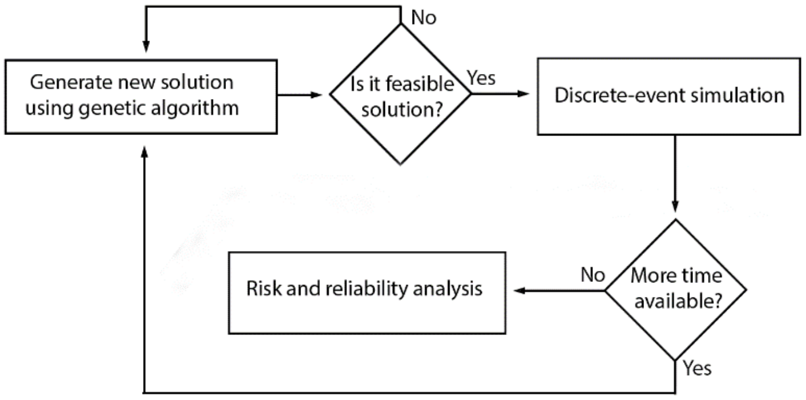
Generate new solution using genetic algorithm
No
Is it feasible solution?
Yes
Discrete-event simulation
No
More time available?
Yes
Risk and reliability analysis

Figure 1. The logic behind the simulation-driven approach

Unequivocally, the most significant drawback of such a combined approach is that the solutions are not expected to be optimal. However, real-life stochastic optimization is commonly NP-hard in nature, thus, the combination of simulation with metaheuristics seems to be a tempting alternative for practical tasks, since such an approach provides a relatively simple and flexible method to deal with complex problems in reasonable computing times.

3. The Simulation Description

First of all, the proposed method requires designing a simulation that corresponds to the real system with a sufficient degree of accuracy. As it is already mentioned, such a simulation with stochastic parameters will play the role of an objective function. Thus, an optimization process will be reduced to the search of the best simulation adjustments. The inventory theory at its current stage has developed a significant mathematical foundation for solving problems related to the determination of the optimal inventory policy (Zipkin, 2000). The most suitable model among considered is the model of Hopp and Spearman (2008). It is also worth noting that several distinguishing features were taken from "lost sales (r, Q) inventory control model" (Kouki *et al.*, 2015), classical (r, Q) models of Bookbinder and Cakanyildirim (1998) and Bijvank and Vis (2011). The considered model makes several assumptions. Firstly, unfulfilled demands are defined as a lost opportunity and no backlog shall be fulfilled later. Secondly, demand size, demand frequency and replenishment lead time are continuous random variables. Thirdly, a product of a particular type is replenished by an individual supplier.

Discrete-event simulation paradigm is chosen in order to take into account random components without a dramatic increase in system complexity at the computational level. The simulation of this kind was successfully performed and analysed in several recent articles by Alizadeh *et al.* (2011), Min and Lindu (2016) and Sinaga *et al.* (2016). Unlike in continuous simulation, system dynamics is not unceasingly tracked during the simulation time. Discrete-event simulation contains a list of events, such that each event takes place at a particular instant of time altering the state of the system. It is important to emphasize that there are no changes in the system between consecutive events. That is why, the simulation laps in time from previous event to the next one and runs much faster saving precious computational resources (Fig. 2). Each event is scheduled according to preliminary generated time t_n and executes sequentially. Generated time is appended to a time vector $T = (t_0, t_1 \dots t_n)$, which may be interpreted as a clock or a time-counter.

The total inventory assortment corresponds to the set of products P, such that each product $p_i \in P$. The storage capacity allocation is the first priority task. Presuming that I_{max} is the total storage capacity, we may declare B as a vector of individual storage capacities assigned for each product p, such that $B = (\beta_1, \beta_2, \dots \beta_{|P|})$ and $\sum_{i=1}^{|P|} \beta_i = I_{max}$. The simulation begins with an initial inventory level of I_p at t_0. During the simulation, emerging demands $x_{p,t}$ are satisfied and the stock level declines gradually. If the stock level falls below a certain threshold (reorder point) $r_{p,t}$, the inventory places a new order $y_{p,t}$ to refill the stock. Assuming that $x_{p,t}$ cannot exceed the corresponding inventory capacity β_p, an inventory level at a particular moment of time equals to an inventory level in previous moment subtracting received demand and adding an order that was placed at order time $t - L$ Eq. (4). Where L is the replenishment lead time, the time it will take to deliver the product from a supplier to the warehouse. It is also worth to note that such a model aims to represent an inventory under some sort of just-in-time policy, thus, the order size $y_{p,t-L}$ (Q in classical (r, Q) models) equals to the corresponding maximal inventory capacity β_p subtracting the difference between the current inventory level $I_{p,t}$ and adjusted safety-stock SS_p Eq. (5). In the proposed model, a new reorder point r_p is recalculated after each replenishment Eq. (6). Where $\theta_{p,t-L}$ stands for a mean demand during the replenishment lead time and SS_p is a value of the corresponding safety-stock. Based on that, the number of arisen backorders $O_{p,t}$ may be determined as the step function Eq. (7).

$$I_{p,t+1} = I_{p,t} - x_{p,t} + y_{p,t-L}, \tag{4}$$

$$y_{p,t} = \begin{cases} \beta_p - (I_{p,t} - SS_p), if\ I_{p,t} > SS_p \\ \beta_p, if\ I_{p,t} \le SS_p \end{cases}, \tag{5}$$

$$r_{p,t} = \theta_{p,t-L} + SS_p, \tag{6}$$

$$O_{p,t} = \begin{cases} 0,\ if\ x_{p,t} \le I_{p,t} \\ x_{p,t} - I_{p,t},\ if\ x_{p,t} > I_{p,t} \end{cases}. \tag{7}$$

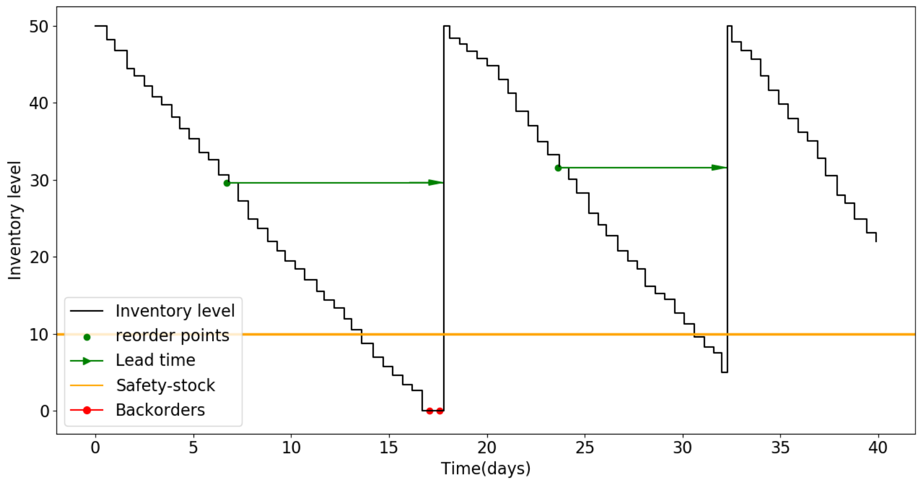

Figure 2. The example of an inventory path

Discrete-event simulation of such models is simple enough and can be performed by the iterative algorithm (Fig. 3).

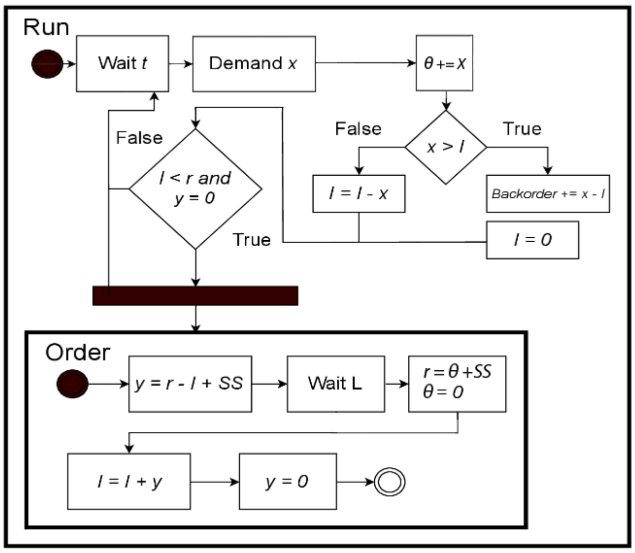

Figure 3. The logic behind the simulation

Each product in an assortment has a different market price and thus a different backorder cost b_p. Likewise, unit costs of storage and shipping, h_p and l_p respectively, vary depending on product's properties and subtleties of handling. Thereby, the total cost function for each product is the sum of the products of unit costs on number of units shipped, stored or backordered respectively Eq. (8).

$$TC_p = l_p \sum_{i=0}^{t} y_{p,t} + h_p \sum_{i=0}^{t} I_{p,t} + b_p \sum_{i=0}^{t} O_{p,t}. \quad (8)$$

According to Eq. (5), the overflow ($\sum_{i=1}^{|P|} I_i > \sum_{i=1}^{|P|} \beta_i$) may occur. Such a case may be taken into account by declaring a specific cost s related to the unit overflow and tracing the overflow level Eq. (9). In real world, such a cost corresponds to the warehouse outsourcing or reverse logistics.

$$F_t = \begin{cases} 0, & if \ \sum_{i=1}^{|P|} I_i \le \sum_{i=1}^{|P|} \beta_i \\ \sum_{i=1}^{|P|} I_i - \sum_{i=1}^{|P|} \beta_i, & if \ \sum_{i=1}^{|P|} I_i > \sum_{i=1}^{|P|} \beta_i \end{cases}. \quad (9)$$

In this regard the total costs function for an inventory as a whole is $\sum_{p=1}^{|P|} TC_p + s \sum_{i=0}^{t} F_t$.

4. The Optimization Procedure

4.1. The essence of genetic algorithms

The genetic algorithm is a stochastic search technique that mimics the evolutionary phenomena of natural selection, namely, the chromosome inheritance and darwinistic struggle for survival. The genetic algorithm was invented and firstly introduced by Holland (1975). To date, genetic algorithms have been successfully implemented in logistics and supply chain management (Altiparmak *et al.*, 2006) and (Yeh and Chuang, 2011). It is expected that the algorithm converges to the optimum in several generations (Pasandideh and Niaki, 2008). According to the research (Man *et al.*, 1996), genetic algorithms are quite simple to design, however, their behaviour is hard to predict and understand, especially, why genetic algorithms frequently succeed at providing near-optimal solutions in real-world stochastic problems. Conventionally, such phenomena are attempted to be explained by the building-block hypothesis. The hypothesis says that genetic algorithm looks for a near-optimal solution by the juxtaposition of the schemes with high fitness (the building block). The highly-fit strings (or other data structures) are sampled, crossed over and resampled to form new even fitter strings. Working with these building blocks, we manage to reduce the complexity of the problem. Namely, instead of building strings with good fitness by trying every feasible combination, algorithm improves strings step by step from the partially best solutions of the previous samplings.

In order to apply genetic algorithm, the following initial parameters are required:

- Population size (N) – the number of chromosomes in each generation;
- Crossover rate (P_c) – the probability of executing a crossover operator;
- Mixing ratio (P_u) – the probability for each attribute to be exchanged;
- Mutation rate (P_{m1}) – the probability of executing a mutation operator 1;
- Mutation rate (P_{m2}) – the probability of executing a mutation operator 2;
- Mutation step (δ) – the gene-multiplier used by the mutation operator 2;
- Tournament size (t).

The pivotal steps involved in the applied genetic algorithm are described in pseudocode (Luke, 2015):

```
$N$ ← desired population size
$P$ ← {}
for $N$ times do
  $P$ ← $P$ ∪ {new random individual}
Best ← □
repeat
  for each individual $P_i \in P$ do
    AssesFitness($P_i$)
    if $Best$ = □ or Fitness($P_i$) > Fitness($Best$) then
      $Best$ ← $P_i$
  $Q$ ← {}
  for $N/2$ times do
    Parent $P_a$ ← SelectWithReplacement($P$)
    Parent $P_b$ ← SelectWithReplacement($P$)
    Offspring $C_a$, $C_b$ ← Crossover(Copy($P_a$), Copy($P_b$))
    $Q$ ← $Q$ ∪ {Mutate($C_a$), Mutate($C_b$)}
  $P$ ← $Q$
until time runs out
return $Best$
```

4.2. Chromosome representation and fitness function

Practically, genetic algorithm is quite efficient in cases of large search space with lack of knowledge on the structure of the fitness function. The stochastic inventory optimization problem undoubtedly belongs to this domain. Moreover, in cases of high stochasticity (Fig. 4), it becomes difficult to apply some traditional optimization techniques.

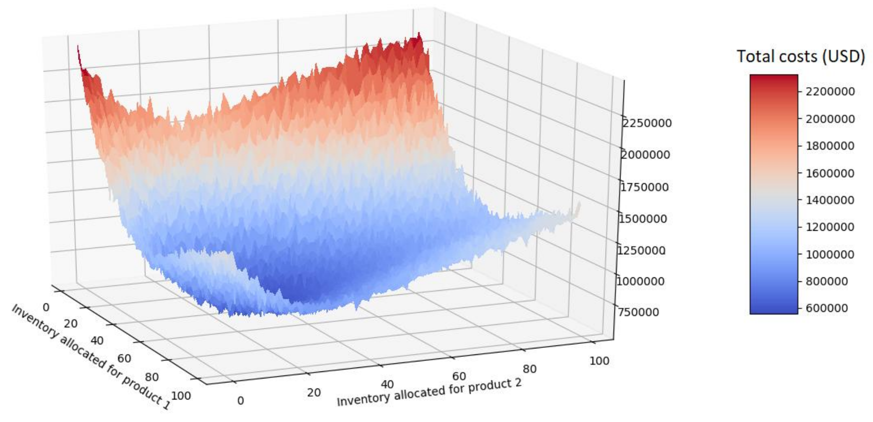

Figure 4. Total costs function of the 2-product stochastic inventory model with the constant safety-stock parameter

Genetic algorithm is quite famous as a problem-independent approach, nevertheless, the chromosome representation is a critical issue. Applying genetic algorithm to the inventory optimization problem under consideration, we are looking for such adjustments to simulation parameters: storage-resources allocation $B = (\beta_1, \beta_2, \ldots \beta_{|P|})$ and corresponding safety-stock levels $SS = (SS_1, SS_2, \ldots SS_{|P|})$ that lead to the best fitness. The chromosome may be encoded as a $|P|$ size list of integers $\vec{v} = (\beta_1, SS_1, \beta_2, SS_2, \ldots \beta_{|P|}, SS_{|P|})$. In such a list each odd element stands for the inventory capacity allocated to each product p and each even element represents adjusted safety-stock level for the corresponding product p (Fig. 5).

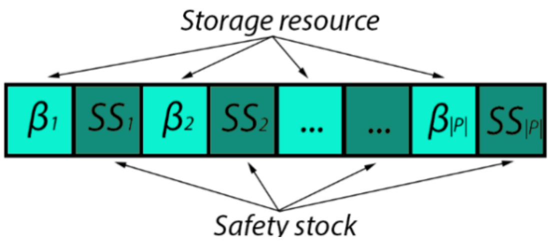

Figure 5. Chromosome representation

In such a simulation-driven approach, fitness function is evaluated by sequential runs of several simulations. In this case, fitness is the mean value of total costs calculated in several sequential simulation's runs. with the same parameters. We are looking for such parameters that lead to the minimal mean value of the total cost function Eq. (10) satisfying the constraints Eqs. (11, 12).

$$\min_{a \in \mathbb{A}} E[\textstyle\sum_{p=1}^{|P|} TC_p(a)], \tag{10}$$

$$\textstyle\sum_{i=1}^{|P|} \beta_i \leq I_{max}; \ \forall i = 1, 2, 3, \ldots, |P|, \tag{11}$$

$$SS_i \leq \beta_i; \ \forall i = 1, 2, 3, \ldots, |P|. \tag{12}$$

In case the solution does not satisfy constraints Eqs. (11, 12) the fitness will take extremely high values, due to infeasibility of such a solution. During the optimization procedure, such individuals (candidate solutions) will have only an insignificant chance to pass to the next generation.

It is pointing out that a suitable chromosome representation for the particular problem domain is an extremely important task, since a good choice will make the search faster and easier by restricting the search space. However, it is tremendously important to keep in mind that the crossover and mutation operators must take into account the design (especially datatype) of the chromosome. It is important to emphasize that in the considered problem a non-binary chromosome representation was chosen.

As it is mentioned in Stack Exchange (2017), the main reason why binary representation is the most frequent is the simplicity to implement and popularity in academic papers. Moreover, binary chromosome representation is usually space-efficient, that is why it was so popular in times, when memory was a serious problem. However, in real-world problems, it becomes common to create a genotype representation that corresponds to the considered problem with a high degree of accuracy.

4.3. Crossover and mutation

Crossover is the distinguishing operator of the genetic algorithm. Basically, it is a process of taking two parent solutions and production of offspring solutions in order to get a new, potentially better one. Crossover is used to vary chromosomes from one generation to the next. In order to solve the stochastic inventory problem, the uniform crossover is proposed (Fig. 6).

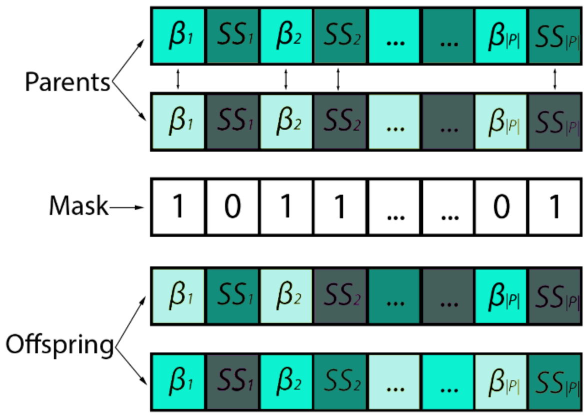

Figure 6. Uniform crossover representation

In the uniform crossover individual genes in the chromosome are compared between two parents and swapped with the fixed mixing ratio P_u. Uniform crossover is chosen for three main reasons. Firstly, since genes in the chromosome correspond to different simulation parameters *SS* and *B*, we seek a way to keep odd and even genes separated. Secondly, the uniform crossover is an efficient way to avoid the premature convergence (Michalewicz, 1996). Lastly, Williams and Crossley provided an empirical evidence (1998) that the uniform crossover is an exploratory approach in comparison to the traditional exploitative one, as the result, it becomes more efficient in more complete search by maintaining the exchange of valuable information.

```
$P_u$ ← probability of swapping values
$\vec{v}$ ← first vector $\langle v_1, v_2, \dots, v_n \rangle$ to be crossed over
$\vec{w}$ ← second vector $\langle w_1, w_2, \dots, w_n \rangle$ to be crossed over
for i in range from 1 to length of the vector do
    if $P_u \geq$ random number in range (0.0, 1.0) then
        swap the values of $v_i$ and $w_i$
return $\vec{v}$ and $\vec{w}$
```

Besides, genetic algorithm requires a mutation operator to perform the optimization. Taking into account the particularities of chromosome encoding, it is proposed to apply two different mutation operators (“mild” and “radical”). The radical mutation is applied in order to prevent the premature convergence (otherwise population may get stuck in local optima). In radical mutation we replace gens in the chromosome by a new integer number in a feasible range (0, I_{max}) with the probability P_{m1}.

```
$P_{m1}$ ← probability of replacing the value
$\vec{v}$ ← vector
for i in range from 1 to length of $\vec{v}$ do
    if $P_m \geq$ random number in range (0.0, 1.0) then
        $v_i$ ← random integer in feasible range
return $\vec{v}$
```

On the other hand, mild mutation is applied to accelerate convergence. The mild-mutation operator alters genes in the chromosome with the probability P_{m2} by multiplying them on some (relatively small) step δ rounding to the nearest integer after that.

```
$P_{m2}$ ← probability of altering the value
$\vec{v}$ ← vector
for $i$ in range from 1 to length of $\vec{v}$ do
  if $P_m \geq$ random number in range (0.0, 1.0) then
    $v_i$ ← round($v_i * \delta$)
return $\vec{v}$
```

4.4. Selection

It is concluded by Miller and Goldberg (1995) that tournament selection is an efficient and robust mechanism for working with imperfect (noisy) fitness functions. Tournament selection runs several "tournaments" among *t* individuals (chromosomes) randomly chosen from the population. The fittest individual in each tournament is selected for the following crossover. Since weak individuals have relatively a small chance to be selected in large tournaments, it is quite important to find the optimal tournament size *t*. Tournament Selection can be programmed by the extremely simple algorithm:

```
P ← population
t ← tournament size, t ≥ 2
Best ← randomly picked individual from P with replacement
for i in range from 2 to t do
   Next ← randomly picked individual from P with replacement
   if Fitness(Next) > Fitness(Best) then
      Best ← Next
return Best
```

Tournament selection has several significant benefits over alternative selection methods, namely, it is both simple and efficient to code, it works with parallel architectures and, lastly, it may be easily adjusted.

5. The Numerical Example

Consider an example of the six-product inventory control system that operates under just-in-time policy. There is a retailer selling products of 6 types that are replenished by an individual supplier. Products of all six types share a common storage with a limited capacity of 150 pallets $\sum_{i=1}^{6} \beta_i \leq 150$. Each type of product has a unique triangular distribution (Kotz and Van Dorp, 2004) for both demand size and replenishment lead time and exponential distribution for demand interarrivals (Table 1.).

Table 1. The initial data

Storage costs (USD)	Shipping costs (USD)	Backorder costs (USD)	Lead time (days)	Demand size (pallets)	λ of demand interarrivals (days)
7.2	3.5	730	(7.5, 9.0, 11.5)	(0.3, 0.85, 1.2)	2.0
6.32	3.2	650	(3.5, 5.5, 7.5)	(0.7, 1.45, 2.1)	1.11
4.32	2.7	350	(4.2, 5.3, 9.5)	(0.45, 0.75, 1.3)	1.43
5.5	3.0	510	(3.5, 4.5, 7.5)	(0.43, 0.65, 2.3)	2.86
8.2	4.3	900	(1.3, 2.2, 3.0)	(0.09, 0.15, 0.5)	0.021
4.5	3.9	270	(1.5, 2.0, 2.9)	(0.26, 0.34, 0.43)	0.9

We apply given adjustments and execute 60-days simulation of the inventory control system. The simulation is designed in “SimPy”, process-based discrete-event simulation framework on standard Python 3.6 (Scherfke, 2014). We also use evolutionary computation framework “DEAP” to develop the genetic algorithm with the required operators. DEAP is chosen, because it works perfectly with parallelisation mechanisms and keeps data structures transparent (Fortin *et al*., 2012).

The algorithm has successfully converged at the optimum in 122 generations. The optimal solution is represented by the chromosome $\vec{v}$ = (30, 4, 17, 2, 24, 4, 41, 5, 15, 1, 13, 1) with the expected total costs of $E[\sum_{p=1}^{6} TC_p(\vec{v})] = 2828835$ USD. The fittest chromosome $\vec{v}$ stands for the following simulation adjustments: storage-resources allocation B = (30, 17, 24, 41, 15, 13) pallets with the corresponding safety-stock levels SS = (4, 2, 4, 5, 1, 1) pallets. The pivotal advantage of the involved discreet-event simulation is a possibility to perform risk and reliability analysis. Additionally, such an approach allows the researcher to plot the inventory dynamics in details and easily spot existing bottlenecks or system vulnerabilities.

Since the parameters of a genetic algorithm (especially a population size N and a tournament size t) have tremendous impact on convergence speed and a probability of premature convergence, it is very important to find a balance between search speed and premature convergence prevention (Fig. 7).

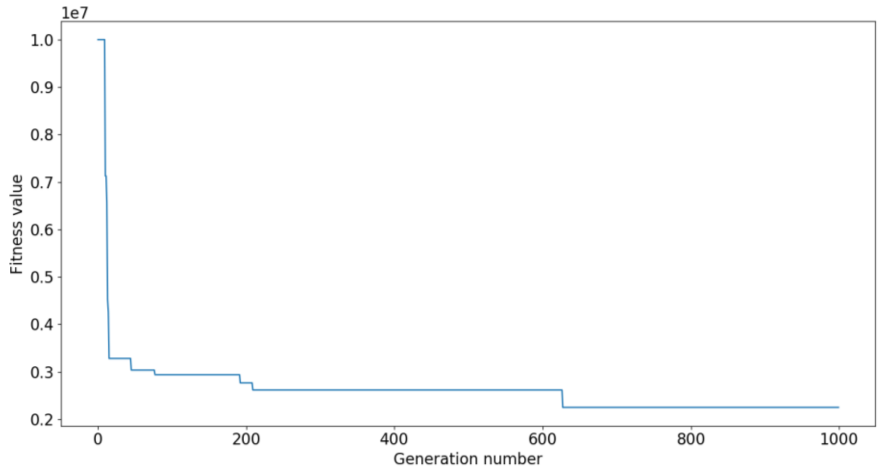

Figure 7. The example of convergence path

Parameters with such a balance may be found empirically (Table 2). According to the Table 2, we can conclude that even relatively insignificant alterations in parameters of the genetic algorithm noticeably affect the convergence speed. Furthermore, some unsuccessful settings may result a premature convergence.

Table 2. The parameters of the genetic algorithm

№	Pop. size (*N*)	Cross. rate (*Pc*)	Mix. ratio (*Pu*)	Mut. rate (*Pm1*)	Mut. rate (*Pm2*)	Mut. step (δ)	Tour. size (*t*)	Fitness value	Conv. speed
1	120	0.3	0.1	0.025	0.1	1.1	2	4437934	43
2	130	0.3	0.2	0.05	0.05	1.15	4	2828811	198
3	140	0.4	0.4	0.025	0.05	1.2	5	2828901	274
4	150	0.4	0.3	0.05	0.05	1.05	3	2828835	122
5	160	0.5	0.5	0.05	0.025	1.25	6	2828622	334
6	170	0.5	0.6	0.1	0.025	1.3	7	2828873	219

Variation of parameters, among other things, is a reliable way to verify a preliminary solution, to make sure that the algorithm has not converged at a local optimum or even to a random point, unfortunately, it is not a panacea.

6. Risk and Reliability Analysis

The remarkable advantage of a simulation-driven approach is the possibility to utilize risk and reliability analysis in the decision-making process. Since system under consideration is characterized by a sufficient degree of uncertainty, a decision-maker may be interested not only in optimization of the expected value, but also in avoidance of undesirable events, such as backorders and overflows. Even a very accurate model may miss some of the subtleties. Furthermore, some undesirable events may be slightly underestimated in a cost function. For example, in real world backorder induces not only loss of opportunity (potential profit), but also loss of business reputation and image, which is extremely complicated to assess monetary. Similarly, abrupt storage-overflow involves either warehouse outsourcing or reverse logistics, an extra difficulty in terms of management. Following the described approach, an expert may eventually choose the most suitable solution among several nearly-optimal candidates based on specific criteria, for instance, the likelihood of a backorder to arise during the replenishment (Fig. 8).

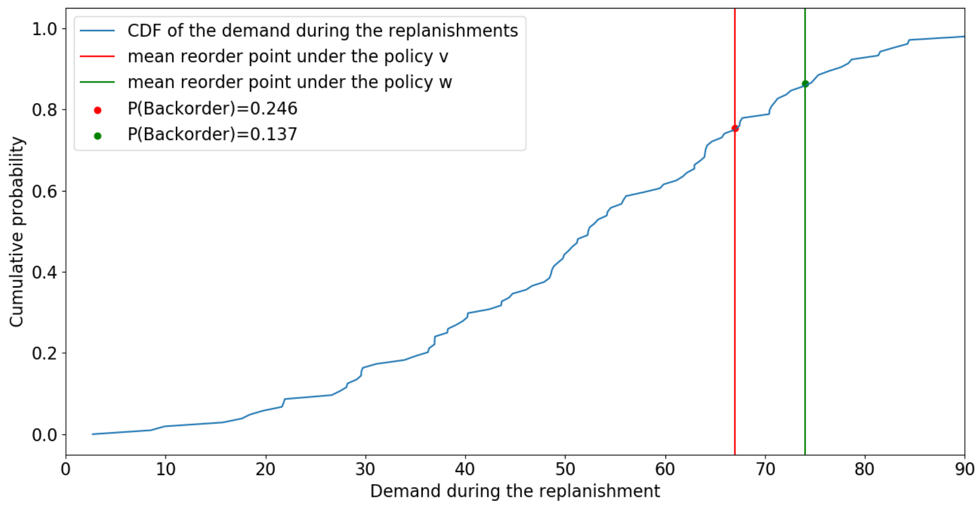

Figure 8. Backorder risk comparison of two candidate-solutions

Besides, risk-averse decision maker will be, most likely, interested in the solution with a smaller standard deviation over the riskier solution with a slightly better expected value (Juan *et al.*, 2015).

7. Conclusion

Summarizing, the proposed optimization technique is a simple to design and computationally efficient approach to find nearly-optimal inventory policy in stochastic multi-product inventory systems. The combination of discrete-event simulation and genetic algorithm provides a flexible method to solve complex problems with lack of knowledge on the structure of the objective function. Furthermore, the discrete-event simulation paradigm takes into account random components. Besides, the key advantage of such a simulation-driven approach is the possibility to trace inventory dynamics in details and utilize risk and reliability analysis in a decision-making process.

The research also concludes with a statement that the non-binary chromosome encoding works properly in combination with uniform crossover and two mutation operators. Such a design provides a fine balance between convergence speed and likelihood of premature convergence. There are still several minor problems to solve, such as the program-optimization of both the simulation and genetic algorithm. Moreover, it is crucially important to test the proposed approach on problems with higher dimension and compare it to alternative metaheuristic techniques. These issues are waiting to be deeply explored in a future research.

Acknowledgements

This paper is based on the research and work that has been partly conducted in the framework of the ALLIANCE project (http://alliance-project.eu/), which has received funding from the European Union's Horizon 2020 Research and Innovation Programme. The authors would like to thank both the consortium of the project and the European Commission.

References

1. Alizadeh, M., Eskandari, H., Sajadifar, S. and Geiger, C. (2011) Analysing a stochastic inventory system for deteriorating items with stochastic lead time using simulation modelling. In: *Proceedings of the 2011 winter simulation conference.* Winter Simulation Conference, pp. 1650-1662.
2. Altiparmak, F., Gen, M., Lin, L. and Paksoy, T. (2006) A genetic algorithm approach for multi-objective optimization of supply chain networks. *Computers & industrial engineering*, 51(1), 196–215. DOI:10.1016/j.cie.2006.07.011

3. Bijvank, M. and Vis, I.F.A. (2011) Lost-sales inventory theory: A review. *European Journal of Operational Research*, 215(1), 1 – 13. DOI:10.1016/j.ejor.2011.02.004
4. Bookbinder, J.H. and Cakanyildirim, M. (1999) Random lead times and expedited orders in (Q, r) inventory systems. *European Journal of Operational Research,* 115(2), 300–313.
5. Brandel, W. (2009) Free Up Cash! Inventory optimization save working capital in tough times. *Computerworld.* [online] www.computerworld.com. Available at: https://www.computerworld.com/article/2549533/it-industry/free-up-cash [Accessed 9 Mar. 2018]
6. Fortin, F.A., Rainville, F.M.D., Gardner, M.A., Parizeau, M. and Gagné, C. (2012) DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research,* 13(Jul), 2171–2175.
7. Holland, J.H. (1975) *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.
8. Hopp, W.H. and Spearman M.L. (2008) *Factory Physics.* Waveland Press.
9. Juan, A.A., Faulin, J., Grasman, S.E., Rabe, M. and Figueria, G. (2015) A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspective,* 2, 62–72. DOI:10.1016/j.orp.2015.03.001
10. Juan, A.A., Grasman, S.E., Caceres-Cruz, J. and Bektaş, T. (2014) A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs. *Simulation Modelling Practice and Theory,* 46, 40–52. DOI:10.1016/j.simpat.2013.11.008
11. Kotz, S. and Van Dorp, R.J. (2004) *Beyond Beta: Other Continuous Families of Distributions with Bounded Support and Applications.* World Scientific.
12. Kouki, C., Jemai, K. and Minner, S. (2015) A Lost Sales (r,Q) Inventory Control Model for Perishables with Fixed Lifetime and Lead Time. *Int. J. Production Economics,* 168, 143–157.
13. Luke, S. (2015) *Essentials of Metaheuristics. A Set of Undergraduate Lecture Notes.* Second Edition, 2.2, pp. 31-55.
14. Man, K.F., Tang, K.S. and Kwong, S. (1996) Genetic algorithms: concepts and applications [in engineering design]. *IEEE transactions on Industrial Electronics,* 43(5), 519–534. DOI:10.1109/41.538609
15. Michalewicz, Z. (1996) Evolution strategies and other methods. In: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, Heidelberg, pp. 159-177.
16. Miller, B.L. and Goldberg, D.E. (1995) Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3), pp. 193–212. DOI:10.1162/evco.1996.4.2.113
17. Min, Z. and Lindu, Z. (2016) Arena Simulation of Multi-Level Medicine Inventory Control in Hospital Pharmacy. *International Journal of Hybrid Information Technology*, 9(6), pp.283-294.
18. Pasandideh, S.H.R. and Niaki, S.T.A. (2008) A genetic algorithm approach to optimize a multi-products EPQ model with discrete delivery orders and constrained space. *Applied Mathematics and Computation*, 195(2), 506–514. DOI:10.1016/j.amc.2007.05.007
19. Pidd, M. (1998) Computer simulation in management science. ISBN:0471979317.
20. Scherfke, S. (2014) *Discrete-event simulation with SimPy*. OFFIS – Institute for Information Technologie, USA.
21. Stack Exchange (2017) *Why do we use binary encoding when it seems so inefficient?* [online] softwareengineering.stackexchange.com. Available at: https://softwareengineering.stackexchange.com/questions/339705/why-do-we-use-binary-encoding-when-it-seems-so-inefficient/339709 [Accessed 8 Mar. 2018].
22. Subramanian, D., Pekny, J.F. and Gintaras, V.R. (2000) A simulation—optimization framework for addressing combinatorial and stochastic aspects of an R&D pipeline management problem. *Computers & Chemical Engineering*, 24(2-7), 1005–1011.
23. Sinaga, S., Pertiwi, L.S. and Ardian, T. (2016) Inventory Simulation Optimization under Non Stationary Demand. *International Journal of Applied Engineering Research*, 11(1), pp. 524-529.
24. Williams, E.A. and Crossley, W.A. (1998) Empirically-derived population size and mutation rate guidelines for a genetic algorithm with uniform crossover. In: *soft computing in engineering design and manufacturing*. Springer, London, pp. 163-172.
25. Yeh, W.C. and Chuang, M.C. (2011) Using multi-objective genetic algorithm for partner selection in green supply chain problems. *Expert Systems with applications*, 38(4), 4244–4253. DOI:10.1016/j.eswa.2010.09.091
26. Zipkin, P.H. (2000) *Foundations of inventory management*. McGrawHill. ISBN-13:978-0256113792. Zvirgzdiņa, B. and Tolujew, J. (2016) Experience in Optimization of Discrete Rate Models Using ExtendSim Optimizer. In: *9th International Doctoral Students Workshop on Logistics*, June, 2016. Magdeburg, Otto von Guericke University, pp. 95-100.