TWO-PHASE SIMULATION OPTIMISATION PROCEDURE WITH APPLICATIONS TO MULTI-ECHELON CYCLIC PLANNING

Galina Merkuryeva[(a)] and Liana Napalkova[(b)]

Riga Technical University
Department of Modeling and Simulation
1 Kalku Street
LV-1658, Riga, Latvia

[(a)]gm@itl.rtu.lv, [(b)]liana@itl.rtu.lv

ABSTRACT

This paper describes a two-phase simulation-based optimisation procedure that integrates the Genetic Algorithm and Response Surface-based Linear Search algorithm for developing optimal power-of-two replenishment policy in multi-echelon environment during the maturity phase of the life cycle of a product. The problem involves a search in high dimensional space with different ranges for decision variables scales, multiple objective functions and problem specific constraints, such as power-of-two and nested/inverted-nested planning policies. The paper provides illustrative example of the two-phase optimisation procedure applied to generic supply chain network.

Keywords: multi-echelon cyclic planning, genetic algorithm, response surface-based linear search

1. INTRODUCTION

For the last years, there has been increasing attention placed on the performance, design, and analysis of multi-echelon supply chains (Merkuryeva and Napalkova 2007). There are two different approaches used to manage supply chains: *single-echelon* and *multi-echelon* approaches.

The single-echelon approach, which includes continuous review, periodic review, single-unit decomposition, Wagner Within, Silver Meal techniques etc., splits multi-level supply chain into separate stages providing suboptimal solutions.

In contrast to it, the multi-echelon approach, which could be applied to non-cyclic and cyclic policies, considers managing all the echelons in a holistic way and, thus, optimises the global supply chain performance. Compared to non-cyclic policies, which are more preferable from theoretical point of view, *cyclic planning* in multi-echelon environment has more practical benefits, because it provides easy control, reduced administrative costs and safety stocks, and elimination of bullwhip effect (Campbell and Mabert 1991). The main idea of cyclic planning is to use cyclic schedules at each echelon and synchronise them with one another (Merkuryev, Merkuryeva, Desmet and Jacquet-Lagrèze 2007).

Optimisation of multi-echelon cyclic plans refers to the class of multi-objective optimisation problems, which are usually characterized by a large search space of decision variables, conflicting and stochastic objectives etc. (Chen 2003). While there is no a single optimal solution for a number of conflicting objectives, the development of an algorithm, which gives a large number of alternative solutions lying near the Pareto optimal front and tackles the variations of a response generated from the uncertainties in the decision variables and/or parameters, is of great practical value.

Different analytical and mathematical programming methods, such as mixed integer programming, stochastic dynamic programming, network programming, etc., have been developed to define optimal cyclic policies (Campbell and Mabert 1991, Federgruen and Zheng 1993). However, real-world supply chains sometimes cannot expect to get a solution by such methods.

The motivation for current study is to propose a two-phase simulation-based optimisation procedure aimed to find optimal parameters of a multi-echelon cyclic policy for each of supply chain nodes, i.e. replenishment cycles and order-up-to levels, during the maturity phase of the life cycle of a product in order to minimize the sum of ordering, production and inventory costs, respecting fill rate's and cyclic planning constraints, taking into account assumptions of stochastic demand, capacity restriction and backorders.

2. OPTIMISATION PROBLEM STATEMENT

The multi-objective simulation-optimisation problem can be symbolically represented in compact form as (Napalkova and Merkuryeva 2008):

$$\text{Min } \mathrm{E}[F(\mathbf{x})] = \mathrm{E}[f_1(\mathbf{x}),\ldots,f_M(\mathbf{x})], \quad (1)$$
$$\text{subject to: } \mathbf{g}(\mathbf{x}) = \mathrm{E}[\mathbf{r}(\mathbf{x})] \le 0 \text{ and } \mathbf{h}(\mathbf{x}) \le 0,$$

where $E[\cdot]$ is a mathematical expectation; $\mathbf{x} = (x_1,\ldots x_K) \in X$, $\mathbf{f} = (f_1,\ldots,f_M) \in F$; K is a number of decision variables; M is a number of objective functions; X is

called the decision space; F is the objective space; **x** is called a vector of decision variables; **f** is a vector of objective functions; **g** is a vector of stochastic constraints; **h** is a vector of deterministic constraints on the decision variables; **r** is a random vector that represents several responses of the simulation model for a given **x**.

Proceeding from (1), the solution of multi-objective optimisation problem is a vector of decision variables **x** that satisfies all feasible constraints and provides the best trade-off between multiple objectives. To describe objective vector function (1), one could use traditional methods of aggregating multiple objectives into a single objective. The main strength of this approach is a computational efficiency and simple implementation. The weakness is the difficulty to determine a value of the weights that reflect a relative importance of each criterion (Abraham, Jain and Goldberg 2005). Therefore, this paper applies the Pareto dominance concept for finding trade-off solutions.

The trade-off solution $\mathbf{x}^* \in F$ is said to be Pareto optimal (or non-dominated) if there does not exist another $\mathbf{x} \in F$ such that $f_i(\mathbf{x}) \le f_i(\mathbf{x}^*)$ for all criterions $i = 1,\ldots,M$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one criterion j (Abraham, Jain and Goldberg 2005). Finding the Pareto optimal set is a necessary condition for selecting trade-off solutions. Note that this definition of Pareto optimality assumes all the objective functions to be minimized. If some objective function f_i is to be maximized, it is equivalent to minimize the function $-f_i$.

Regarding the problem of cyclic planning within multi-echelon supply chain environment we deal with two objective functions. The first one is to minimize the average total cost represented by sum of inventory holding, production and ordering costs in accordance with the following equation:

$$\text{Min E}[TC] = \sum_{t=1}^{T}\Big(\sum_{j=1}^{J} CP_{jt} + \sum_{i=1}^{I} CO_{it} + \sum_{i=1}^{I} CH_{it}\Big), \quad (2)$$

where TC denotes the total cost, CP_{jt} denotes production cost in process j per period t, CO_{it} is ordering cost at stock point i per period t, and CH_{it} is inventory holding cost at stock point i per period t; I and J correspond to the number of stock points and processes, and T defines the number of periods in the planning horizon.

The second objective function is to maximise customer service requirements specified by the order fill rate.

$$\text{Min E}[FR] = \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{k=1}^{K} QC_{ikt} \Big/ \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{k=1}^{K} D_{kit}, \quad (3)$$

where QC_{ikt} is a fraction of orders provided by stock point i to end-customer k in time period t, D_{kit} is actual demand of end-customer k to stock point i in time period t.

Controlled in optimisation experiments, this performance measure is introduced to avoid unconstrained minimization of the total cost.

The decision variables are replenishment cycles and order-up-to levels, which are considered as discrete and continuous type variables, respectively.

The proposed approach is to integrate (a) the Genetic Algorithm (GA) to guide the search towards an approximate Pareto optimal front and (b) Response Surface-based Linear Search (RSLS) algorithm to improve solutions found by GA. The motivation for selecting these algorithms is the following. As it was mentioned above, the optimisation problem (1) is multi-objective and includes both discrete and continuous type variables. GA is well suited for solving multi-objective combinatorial optimisation problems. RSLS is appropriate to improve existing solutions as it is based on local search approach.

3. THE TWO-PHASE OPTIMISATION PROCEDURE

3.1. Conceptual model

The simulation-based optimisation procedure (Merkuryeva, Napalkova and Hatem 2008) developed to create optimal cyclic schedules in multi-echelon environment consists of the following two phases:

- Phase 1: Optimisation of replenishment cycles by Multi-Objective Genetic Algorithm;
- Phase 2: Optimisation of order-up-to levels by Response Surface-based Linear Search.

3.2. Phase 1: Multi-Objective Simulation-based Genetic Algorithm

The Multi-Objective Simulation-based Genetic Algorithm (MOSGA) is aimed to optimising lengths of replenishment cycles in multi-echelon supply chain networks during the maturity phase of the product life cycle. In this phase, corresponding order-up-to levels are calculated using approximate analytical formulas. Table 1 represents main blocks of MOSGA, which are described below. Flowchart of the proposed algorithm is given in Figure 1.

Table 1: Main blocks of MOSGA

Nr	Block	Description
1	Encoding mechanism	Modified binary encoding
2	Initial population	Random initialisation
3	Fitness assignment	Pareto-based ranking
4	Fitness estimation	Through simulation
5	Penalty function	Artificial increase of total cost
6	Crossover & mutation operators	Uniform crossover One-point mutation
7	Diversity preserving mechanism	Crowding distance
8	Selection strategy	Crowded tourn. selection
9	Elitism strategy	$(\mu + \lambda)$ - selection
10	Termination criterion	Fixed nr of generations

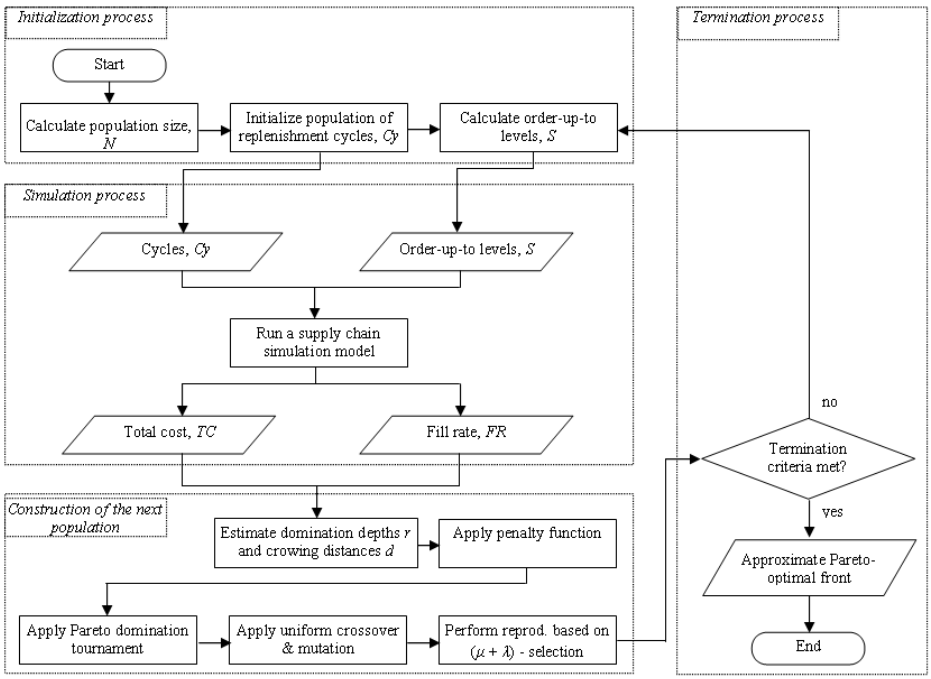

Figure 1: The MOSGA scheme

3.2.1. Encoding mechanism

Encoding mechanism is aimed to codify replenishment cycles for supply chain stock points that have to satisfy power-of-two policy constraints. Here, the cycles are represented by multiples of two. Therefore, the following modified encoding procedure is introduced that allows codifying cycles by the only powers of the base 2.

For stock point i in chromosome *n*:

1. Represent the cycle Cy_{in} as a multiple of two, i.e. 2^p, $Cy_{in} = t*2^p$, where t is a time-step parameter, p is a power of the base 2.
2. Encode a power p to a binary string a_{Ln}, where L is a length of a chromosome.

For instance, the cycle $Cy_{in} = 28$ could be represented as $7*2^2$, where the time-step parameter $t = 7$ days. Then, the power $p = 2$ is encoded to a binary string $a_{2n} = <1, 0>$, i.e. $0*2^0 + 1*2^1 = 2$.

3.2.2. Initial population

Initial population is generated randomly by using uniform distribution, in order to cover the investigated search space. To define a lower bound of the population size N that guarantees both adequate genetic diversity and reasonable simulation processing time, the following D. Goldberg's formula is used:

$$N = 1.65*2^{(0.2*L)} \quad (4)$$

The described below steps are applied to create an initial population.

1. Generate a population $P_N = \{Cy_{11} = 2^{p1}, Cy_{21} = 2^{p2},\ldots, Cy_{IN} = 2^{pN}\}$, where I is a number of stock points in the supply chain, by using "Minimal standard" pseudo-random number generator of Park and Miller.
2. (optional). Sort replenishment cycles subject to nested or inverted-nested policy.
3. Calculate order-up-to levels S_{in} for each stock point i in each chromosome n from the population P_N by using the sequence of analytical approximate formulas described in Napalkova and Merkuryeva (2008).

3.2.3. Fitness assignment and estimation

Fitness is defined based on two objectives represented by performance measures, i.e. the total cost and fill rate that are obtained from simulation experiments. To estimate fitness values of chromosomes, the Pareto-based ranking originally proposed in the *Non-dominated Sorting Genetic Algorithm II* (Deb, Agrawal and Meyariv 2000) is applied.

The example provided below illustrates the concept of the Pareto-based ranking.

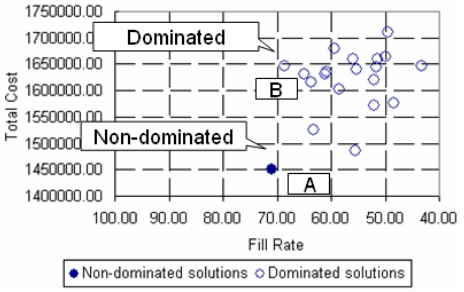

Figure 2: Example of the Pareto-based ranking

The solution represented by the point *A* in Figure 2 is better than the solution in the point *B*, as it gives higher fill rate at lower total cost. This means that the solution *A* is non-dominated and belongs to the first

non-dominated front (domination depth is 1), while the solution *B* is dominated.

The following steps are used for the Pareto-based fitness assignment:

1. Find non-dominated solutions in the entire population and assign them domination depth $r_n = 1$, $n \in R_r$, where R_r is the approximate Pareto optimal set of non-dominated solutions with a domination depth r_n.
2. Temporally exclude non-dominated solutions from the population.
3. Find new non-dominated solutions in the remaining population and assign them domination depth $r_n = r_n + 1$, $n \in R_r$.
4. Repeat Steps 2-3 until all chromosomes are ranked.
5. Reorder solutions according to their fitness values in the increasing sequence, $FITNS = \{fitns_1 \leq fitns_2) \leq ... \leq fitns_N\}$.

3.2.4. Penalty function

Penalty function is introduced to decrease the survival probability of solutions, which provide a fill rate lower than the pre-defined threshold. If a solution has a fill rate lower than threshold, its total cost is artificially increased as follows:

$$F_p: TC*k, \quad (5)$$

where k is a multiplier coefficient.

Here, the fill rate's threshold is defined by 75% and the multiplier coefficient k is equal to 2. In this case, chromosomes with the fill rate below the threshold 75 % are automatically excluded from the population. The value of coefficient k could be also adjusted within the optimisation process.

3.2.5. Crossover and mutation

Crossover and mutation operators provide the exploration and exploitation of a search space. The exploration is used to investigate new and unknown areas in a search space. The exploitation is aimed to making use of knowledge acquired by exploration to reach better positions on the search space. In the proposed GA, the uniform crossover and one-point mutation with probabilities [0.5; 0.8] and [0.01; 0.1], respectively, are introduced.

3.2.6. Diversity preserving mechanism

In order to obtain solutions uniformly distributed over the Pareto front, the diversity preserving mechanism based on a crowding distance is implemented in MOSGA.

The crowding distances d_n are calculated based on the values of total cost and fill rate that are normalized in the interval [0.0; 1.0].

1. For each chromosome n initialize crowding distances $d_n = 0$, $n = 1,\ldots,N$.
2. Sort chromosomes a_{Ln}, $n = 1,\ldots,N$ subject to normalised total cost $f_n^{1,norm}$.
3. Assign $d_1 = \infty$.
4. For each crowding distance d_n, $n = 2,\ldots,N$-1 do Step 5.
5. $d_n = d_n + d^1_{n-1,n+1}$, $d^1_{n-1,n+1} = f_{n+1}^{1,norm} - f_{n-1}^{1,norm}$.
6. Sort chromosomes a_{Ln}, $n = 1,\ldots,N$ subject to normalised fill rate $f_n^{2,norm}$.
7. Assign $d_N = \infty$.
8. For each crowding distance d_n, $n = 2,\ldots,N$-1 do Step 9.
9. $d_n = d_n + d^2_{n-1,n+1}$, $d^2_{n-1,n+1} = f_{n+1}^{2,norm} - f_{n-1}^{2,norm}$.

The main advantage of the crowding approach is that a user doesn't have to define additional parameters, such as, for example, a sharing parameter.

3.2.7. Selection strategy

Crowded tournament selection originally proposed in NSGA-II [14] is used in MOSGA. The main idea of this selection strategy is that a crowded comparison operator is applied to choose the better chromosome in randomly selected ones. Because domination depth is used to assign fitness values, chromosomes with the same domination depth could be selected. In this case, to define the better chromosome, an additional attribute is introduced. In the algorithm, a crowding distance, which is an estimate of the density of solutions surrounding the current solution, is proposed as the additional attribute. Chromosomes with bigger crowding distances have more chances to be selected. Crowding distances of chromosomes, which provide the best value for each objective, are assigned to 999999 that means ∞. As a result, every chromosome in the population has the following two attributes: (1) domination depth and (2) crowding distance. From two solutions the solution with the lower domination depth is preferable. If both solutions have the same depth then the solution with larger crowding distance is preferable.

The crowded comparison operator ($\geq$) is defined as follows:

$$a_{Lj} \geq a_{Lk} \text{ if } (r_j < r_k) \text{ or } ((r_j = r_k) \text{ and } (d_j < d_k)), \quad (6)$$

where d_j and d_k are crowding distances for chromosomes j and k.

3.2.8. Elitism strategy

Elitism strategy is used to avoid the loss of non-dominated solutions during the evolution process. In each generation, an offspring population is added to a parents' population. Domination depths of chromosomes in the combined population are updated. First N solutions are gathered for the next generation, where N is a population size. This strategy is often called as $(\mu + \lambda)$ – selection, where μ and λ assign parents and mating pool, respectively.

The Genetic Algorithm automatically stops the optimisation when the pre-defined number of populations is generated.

3.3. Phase 2: Response Surface-based Linear Search algorithm

The algorithm in the phase 2 is aimed to improve the cyclic planning solution of the Genetic Algorithm received in phase 1 by adjusting analytically calculated order-up-to levels of stock points that could result in decreasing the total cost and/or increasing the end-customers fill rate. It is based on the Response Surface-based Methodology (RSM) applied to simulation optimisation problem (Merkuryeva 2005).

The developed Response Surface-based Linear Search (RSLS) algorithm is based on local approximation of the simulation response surface by a regression type meta-model in a small region of independent factors and integrates linear search techniques for optimising stock points' order-up-to levels. A linear search is a sequential procedure that in each iteration m includes the following main building blocks:

1. Local approximation of the response surface function,
2. Checking the fit of a meta-model,
3. Linear search in the steepest descent direction,
4. Updating the Pareto front.

3.3.1. Local approximation of the response surface function

The total cost is defined as a simulation response and order-up-to levels as input factors. Local approximation of the response surface function in the small region of input factors is performed by using a first-order model. The small region of input factors, or a local search space, is described by a central point, lower and upper bounds.

In iteration m lower t_i^m and upper u_i^m bounds of the local search space, or a region of experimentation, are defined as *proc* % decrease and increase from the central point, respectively:

$$t_i^m = \xi_{i0}^m - proc * \xi_{i0}^m , \; i = 1,...,I , \tag{7}$$

$$u_i^m = \xi_{i0}^m + proc * \xi_{i0}^m , \; i = 1,...,I , \tag{8}$$

where ξ^m_{i0} is a central point of input factor i.

An example of a local search space definition in a set of order-up-to levels is given in Figure 3.

Number of decision variables:	33		
	S_2		
Central point	20718		3469
Lower bound	20510		
Upper bound	20926		
Distance from center point to bound	208		

20510 = 20718 - 208

20926 = 20718 + 208

Figure 3: Example of local search space calculation

To increase the numerical accuracy in estimation of regression coefficients, input factors are coded by the following formula:

$$x_i^m = \frac{\xi_i^m - \xi_{i0}^m}{c_i^m} , \; i = 1,...,I , \tag{9}$$

where x_i^m is a coded input factor i; c_i^m is a distance between the central point and lower (or upper) bound.

Encoding procedures for order-up-to levels are illustrated in Figure 4.

	S_2	S_3	S_4	S_5	S_coded_2	S_coded_3	S_coded_4
1	20926	1393	7247	3434	1	1	1
2	20926	1365	7103	3434	1	-1	-1
3	20926	1393	7103	3434	1	1	-1
4	20510	1365	7103	3434	-1	-1	-1
5	20926				1	-1	-1
6	20510				-1	1	1
7	20510				-1	1	1
8	20926	1393	7103	3504	1	1	-1

20926-> upper bound -> 1

Figure 4: Example of encoding procedure

To make local approximation of a simulation response surface function, the first-order regression meta-model for coded input factors that describes main effects of input factors is used in iteration m:

$$y = b_0^m + \sum_{i=1}^{q} b_i^m x_i^m + \varepsilon^m , \; i = 1,...,I , \tag{10}$$

where b_i^m is a regression coefficient of input factor i; ε^m is a statistical error of a regression model.

In order to estimate meta-model coefficients b_i^m from simulation experiments, the Plackett-Burman experimental design added by simulation replications in the central point is applied. The template for the proposed design has been generated in Minitab statistical software.

Finally, estimates of coefficients of the meta-model (10) are calculated by using well-known method of least squares.

3.3.2. Checking the fit of a meta-model

Lack-of-Fit test is performed to check the adequacy of a regression meta-model and to verify the least-squares method assumptions. Testing lack-of-fit, that gives the determination coefficient close to 1 (100%) and p-value < 0.05 implies the resulted meta-model to be adequate.

3.3.3. Linear search in the steepest descent direction

A linear search is performed within the local search space for order-up-to levels, in the steepest descent direction defined by a vector (b_1^m, b_2^m,…, b_q^m) starting from the central point, where b_1^m, b_2^m,…, b_q^m are coefficients of the simulation meta-model received in iteration m. The increment along the projection of the search direction is calculated only if corresponding regression coefficient is significant (p-value < 0.05). These increments of coded Δx_i^m and decoded Δ_i^m input factor i are calculated as follows:

$$\Delta x_i^m = \frac{-b_i^m}{\max_i \left| b_i^m \right|} , \; i = 1,...,I , \tag{11}$$

$$\Delta_i^m = \Delta x_i^m * c_i^m,\ i = 1,...,I\ . \qquad (12)$$

The values of decoded input factors in the next step of a linear search within iteration m are calculated as follows:

$$\xi_i^m = \xi_i^m + \Delta_i^m,\ i = 1,...,I\ . \qquad (13)$$

The following termination rules are proposed to stop the linear search in iteration m:

1. The simulation response value could not be improved,
2. The search goes outside the pre-defined local search space.

Lower and upper bounds of a new experimental region in iteration $m+1$ are updated by using formulas (7, 8).

3.3.4. Updating the Pareto front

Solutions found during the RSLS are included in the approximate Pareto optimal set initially generated by the Genetic Algorithm. Then all solutions in the resulting Pareto optimal set are reordered according to their fitness values in the increasing sequence.

4. RESULTS AND ANALYSIS

4.1. User interface

The user-interface of MOSGA is developed in MS Excel using ActiveX controls. There are three main user-interface windows, which are used to group the input data, GA and simulation options.

The input data window defines the number of input factors, minimal and maximal values of replenishment cycles, synchronization policy, etc.

The window of algorithm options (Figure 5) describes the number of genes to encode input factors, population size, a type of selection and reproduction strategies, etc.

The window of simulation options describes the number of simulation replication per simulation experiment, length of simulation run warm-up period in hours.

Control buttons allow the user to load the simulation model, run and terminate the optimisation algorithm, as well as calculate the population size based on D. Goldberg's formula (4) for binary-encoded chromosomes in such a way that every solution in the search space is attainable with the crossover genetic operator.

4.2. Input data description

The application itself is aimed to find an optimal cyclic plan of a chemical product, i.e. liquid based raisin, in order to minimise inventory holding, ordering and production costs, and maximise end-customers fill rate. As a test bed, the chemical manufacturing supply chain is used. The main operations occurred in the supply chain network are the following. In the plant CH (see, Figure 6), the raw material is converted to the liquid based raisin. It is then either sourced to direct customers or shipped to the plant DE, where other components are added to make different products. From that plant, the end-products are shipped to different types of customers.

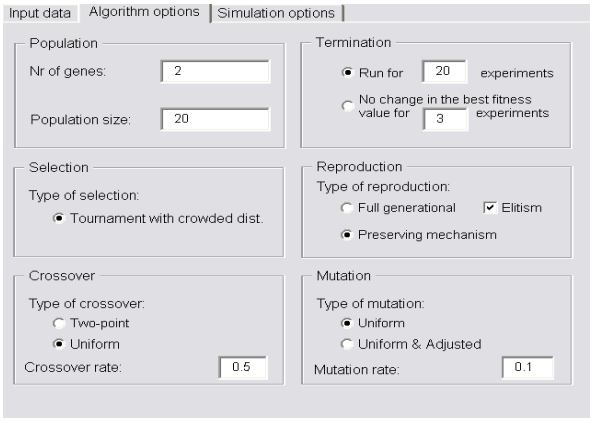

Figure 5: Example of MOSGA user-interface

The ProModel-based simulation model of the above-described supply chain network is generated automatically using a simulation-based optimisation environment presented in Merkuryeva and Napalkova 2007. The end-customer demand is normally distributed; and replenishment cycles are defined according to the power-of-two policy. Cycles are presented in weeks as follows, 7, 14, 28, 56, where 56 days is the maximal cycle that corresponds to one full turn of a "planning wheel". In this business case, specific policies such as nested or inverted-nested ones are not analysed. Initial stocks are equal to order-up-to levels plus average demand multiplied by cycle delays. Stock point 1 has infinite on hand stock and is not controlled by any policy. Backorders are delivered in full.

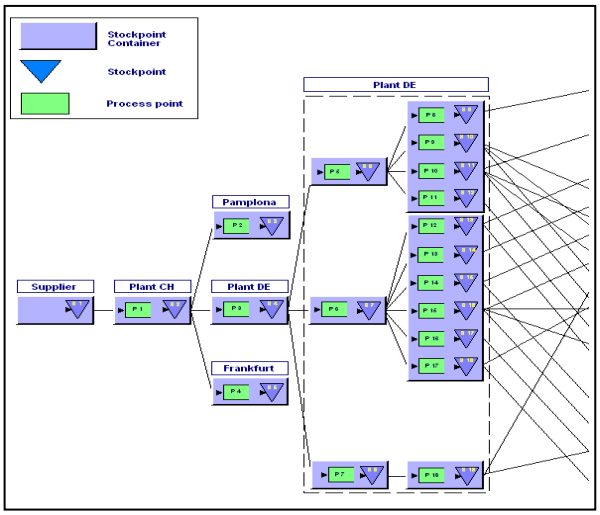

Figure 6: Example of a generic network simulation model

Simulation run length T is equal to 224 periods. The warm-up period is defined by 112 periods. This allows modelling of two full turns of the "planning wheel", i.e. 2*56 periods, during the warm-up period. Number of simulation replications is equal to 5.

4.3. Solutions found by the Genetic Algorithm

To optimise replenishment cycles, GA is executed with parameters summarized in Table 2.

Table 2: Parameters of the Genetic Algorithm

Parameter	Value
Population size	20
Crossover probability	0.5
Mutation probability	0.1
Tournament size	2

The algorithm works with 33 decision variables, which are assigned to network stock points. When the number of generated population is equal to 16, the algorithm is terminated.

Figures 2 and 7 show solutions received from initial and final populations. It could be noticed that at the beginning the approximate Pareto optimal set includes a single non-dominated solution. However, during the evolution process the diversity of the approximate Pareto optimal set is increased, and the final population includes seven non-dominated solutions (Figure 7).

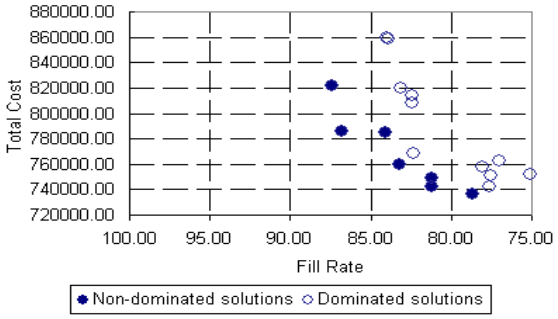

Figure 7: Final population mapped in the objective space

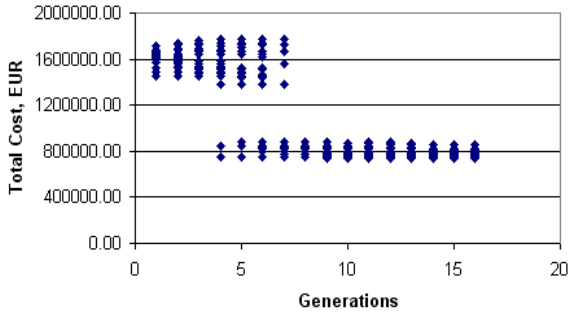

Figure 8: The Genetic Algorithm's convergence subject to Total Cost

Figures 8 and 9 illustrate execution of the Genetic Algorithm. The total cost and fill rate of parent chromosomes are plotted against the generation step. The algorithm makes quick progress in the beginning of the evolution that is typical for Genetic Algorithms. Then, there are phases when it hits the local optimum before mutations further improve its performance.

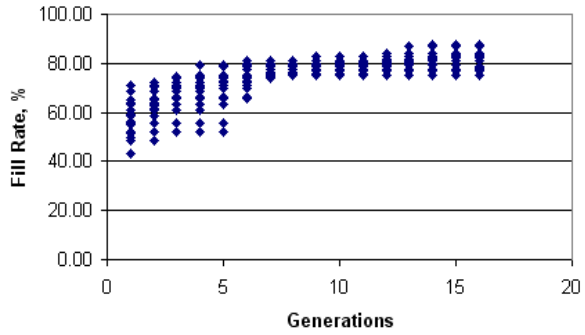

Figure 9: The Genetic Algorithm's convergence subject to Fill Rate

4.4. Solutions adjusted by the Response Surface-based Linear Search algorithm

For each non-dominated solution received in phase 1, stock points' replenishment cycles are fixed and their order-up-to levels are optimised by RSMS algorithm.

Next, the Pareto front generated by GA is updated by adding solutions found within the RSMS (Figure 10).

As a result, we get that these solutions dominate the solutions received in the phase 1 and could substitute them in the final approximate Pareto front (Figure 11).

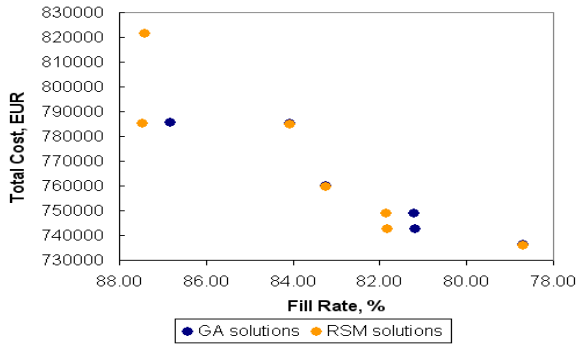

Figure 10: Solutions of MOSGA and RSMS algorithms

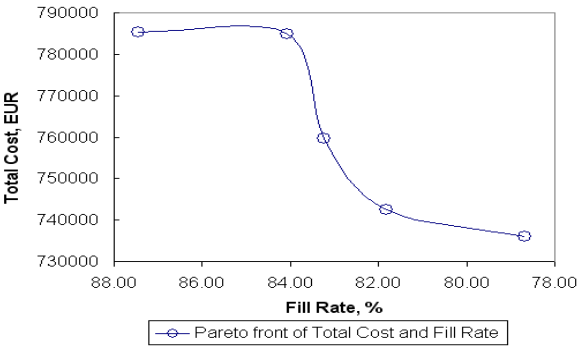

Figure 11: The final approximate Pareto front

5. CONCLUSIONS

The paper presents the simulation-based optimisation algorithm developed in order to define the optimal

lengths of cycles and stock point order-up-to levels during the maturity phase of the life cycle of a product. It integrates the Genetic Algorithm used to optimise replenishment cycles and Response Surface-based Linear Search Algorithm that allows adjusting order up-to levels of stock points when replenishment cycles are fixed. The algorithm is applied to optimise parameters of cyclic plans in multi-echelon supply chain. The achieved results show the efficiency of the developed optimisation procedure. Future research will focus on the performance analysis of the developed procedure and algorithms in solving new problem instances.

ACKNOWLEDGMENTS

The presented research is supported by the European Social Fund within the National Programme "Support for the carrying out doctoral study programmes and post-doctoral researches" project "Support for the development of doctoral studies at Riga Technical University".

The authors would like to thank Jonas Hatem from Möbius Ltd. for helpful comments and discussions.

REFERENCES

Abraham, A., Jain, L., Goldberg, R. 2005. *Evolutionary Multiobjective Optimisation: theoretical advances and applications*. United States of America: Springer.

Campbell, G.M., Mabert, V. A., 1991. Cyclical Schedules for Capacitated Lot Sizing with Dynamic Demands. *Management Science*, pp. 409 – 427.

Chen, J.-H., 2003 Theoretical Analysis of Multi-Objective Genetic Algorithms – Convergence Time, Population Sizing and Disequilibrium. *Report for IEEE NNS Walter Karplus Research Grant*.

Deb, K., Agrawal, S., Meyarivan, T., 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimisation: NSGA-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, Springer, Lecture Notes in Computer Science No. 1917, pp. 849 - 858.

Federgruen, A., Zheng, Y-S., 1993. Efficient Algorithms for Finding Optimal Power-Of-Two Policies for Production/Distribution Systems with General Joint Setup Costs. *Operation Research*, Vol. 43, No. 3, pp. 458 – 470.

Merkuryev, Y., Merkuryeva, G., Desmet, B., Jacquet-Lagrèze, E., 2007. Integrating Analytical and Simulation Techniques in Multi-Echelon Cyclic Planning. *Proceedings of the First Asia International Conference on Modelling and Simulation (AMS 2007)*, pp. 460 – 464. March 27 – 30, Phuket, Thailand.

Merkuryeva, G., Napalkova, L., 2007. Development of simulation-based environment for multi-echelon cyclic planning and optimisation. *Proceedings of 6th EUROSIM Congress on Modelling and Simulation*, paper ID 452. Ljubljana, Slovenia.

Merkuryeva, G., Napalkova, L., Hatem, J., 2008. *Deliverable D2.1.5: Report on the response surface based meta-modelling optimisation algorithm for defining lengths of cycles during maturity phase of the life cycle of a product*. Eclips project.

Merkuryeva, G., 2005. Response surface-based simulation metamodelling methods with applications to optimisation problems. Chapter 15. *Supply chain optimisation Product / Process Design, Facility Location and Flow control* / Eds. A. Dolgui, J. Soldek and O.Zaikin, Springer, pp. 205 – 215.

Napalkova, L., Merkuryeva, G., 2008. Theoretical Framework of Multi-Objective Simulation-Based Genetic Algorithm for Supply Chain Cyclic Planning and Optimisation. *Proceedings of 10th International Conference of Computer Modelling and Simulation.* Cambridge, England.

Veldhuizen, V. A., 1999. *Multi-objective Evolutionary Algorithms: Classifications, Analysis and New Innovations*. Thesis (PhD). Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

AUTHORS BIOGRAPHY

GALINA MERKURYEVA is Professor at the Department of Modelling and Simulation, Riga Technical University. She holds two degrees: DSc and Dr.sc.ing. G. Merkuryeva has professional interests and experiences in discrete-event simulation, simulation metamodeling and optimisation, simulation-based training, and supply chain simulation and decision support.

LIANA NAPALKOVA holds her MSc degree in Computer Science from Riga Technical University (2006). Currently, she is a PhD student at RTU Department of Modelling and Simulation, and participates in research projects in the logistics field. Her interests focus on the use of simulation-based optimisation techniques for providing competitive advantages in multi-echelon supply chain cyclic planning. Liana Napalkova is a member of the Latvian Simulation Society.