TESTING DIFFERENT EVOLUTION STRATEGY SELECTION STRATEGIES

PAVEL RASKA, ZDENEK ULRYCH

University of West Bohemia, Faculty of Mechanical Engineering
Department of Industrial Engineering

Pilsen, Czech Republic

DOI : 10.17973/MMSJ.2018_03_2017110

e-mal: praska@kpv.zcu.cz

The paper deals with testing different selection strategies of Evolution Strategy on different discrete event simulation models. The models reflect real production systems in industrial companies. We specified different objective functions of models considering the simulated system. All possible solutions and their objective function values were mapped to find the global optimum in the search space. We tested different settings of selection strategies and other Evolution Strategy parameters using a simulation optimizer we developed for simulation optimization. We evaluated these settings by the success of finding the optimum by the optimization algorithms. We also used another evaluation criterion - the difference between the objective function value of the best solution found in the series (replication of optimization experiments with a concrete optimization method setting) and the optimum objective function value.

KEYWORDS

evolution strategy, selection strategies, discrete event simulation models, simulation optimization, objective function

1 INTRODUCTION

The vast majority of industrial companies need to address the problems of optimization of their processes, i.e. production and other activities - logistics, sales, supply, etc. and the appropriate use of their resources, i.e. employees, machines, materials, etc. for these processes, almost every day. The problem is how to quickly find the optimum respecting the specified constraints, because simulated problems are often NP hard problems. A possible answer to this problem is using simulation optimization - experimenting with a discrete event simulation model reflecting real problems. The simulation optimizer tries to find the suboptimal or the optimal feasible solution of the modelled problem by automatically varying the input parameters of the discrete event simulation model. The specified objective function/s represents the quality of the found solution to the modelled problem. The simulation optimizer uses different (heuristic, meta heuristic, etc.) optimization methods to find the optimum of this function (function maximization can be converted to function minimization):

$$\tilde{\mathbf{X}} = \text{argmin}_{\mathbf{X}\in\tilde{X}} F(\mathbf{X}) = \left\{\tilde{\mathbf{X}} \in \tilde{X}; F(\tilde{\mathbf{X}}) \leq F(\mathbf{X}) \forall \mathbf{X} \in \tilde{X}\right\} \quad (1)$$

$\tilde{\mathbf{X}}$ denotes global minimum of the objective function; $F(\mathbf{X})$ denotes objective function value of candidate solution – the range includes real numbers. Objective function represents the aim of simulation optimization; $\tilde{X}$ denotes search space.

This paper deals with the testing of different selection strategies of Evolution Strategy because Evolution Strategy is a very general optimization method which can be used for different types of objective functions. Evolution Strategy has many variants, i.e.σ-self-adaptation (σ SA) [Beyer 2017b], Covariance Matrix Adaptation Evolution Strategy (CMA-ES). [Van Rijn 2016], [Beyer 2017a]

Evolution strategy can be combined with other optimization methods, i.e. neural networks.

2 INFORMATION ENTROPY

This optimum has to respect the specified constraints. We use a Box constraint – search space is limited:

$$\tilde{X}_j = b_j - a_j, a_j \leq b_j \quad (2)$$

$$\tilde{X} = \prod_{j=1}^{n} \tilde{X}_j \quad (3)$$

j denotes the index of decision variable of the simulation model; $\tilde{X}_j$ denotes the length of interval of j-th decision variable; a_j denotes lower bound of the interval of j-th decision variable; b_j denotes upper bound of the interval of j-th decision variable; n denotes dimension of the search space.

We use two different termination criteria to stop the optimization experiment– simulation runs are performed with a concrete optimization method setting to find the optimum of the objective function. The first criterion is Value To Reach- the objective function value of the best candidate solution from all possible candidate solutions in the search space is known. The second termination criterion is the maximum number of simulation runs that can be performed in each optimization experiment. We mapped all the possible solutions in the search space - $\tilde{X}$ – for each discrete event simulation model. The number of all possible solutions in the search space is reduced using the information entropy – Shannon Entropy. [Borda 2011]

The reduction coefficient:

$$\delta = \max\{0{,}1 - \beta \cdot log\tilde{X}\}, \delta \in [0,1] \quad (4)$$

$\tilde{X}$ denotes the size of the search space – the number of all possible solutions in the search space; β denotes the coeficient of search space reduction.

The maximum number of simulation runs that it is possible to perform in each optimization experiment – the second termination criterion:

$$\tilde{X}_H = \lfloor 2^{\delta \cdot log_2 \tilde{X}} \rfloor \quad (5)$$

The curve in Figure 1 shows the dependence of the second termination criterion –the maximum number of simulation runs in the optimization experiment - on the number of the possible solutions in the search space of the discrete event simulation model - $\tilde{X}$.The second termination criterion is not much reduced if the number of the possible solutions in the search space is small. We set the coefficient $\beta = 0.05$ according to our initial optimization experiments.

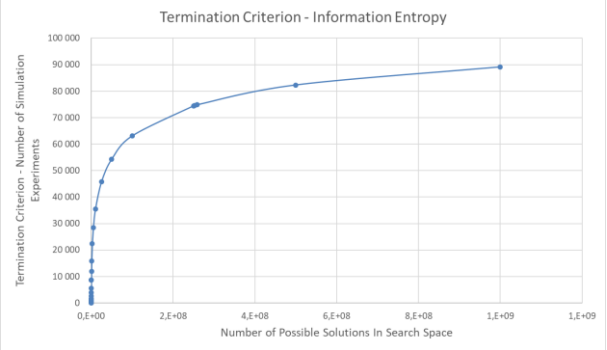

Figure 1. Termination Criterion - Information Entropy

3 TESTED DISCRETE EVENT SIMULATION MODELS

We specified different objective functions according to the aim of the simulation. The quality of the candidate solution represents its objective function value. This value is calculated from the responses of the simulation model after a finished simulation run.

We simulated all feasible candidate solutions of the discrete event simulation model to find the best candidate solution of the search space – the optimum. We also built a database of simulation experiments to increase the speed of simulation optimization. This database contains all the possible settings of the simulation models input parameters and their objective function values. The simulation optimizer searches for the concrete settings of the simulation model input parameters in the local database first before it performs the simulation run with the concrete settings of the simulation model. If it does not find the concrete settings, it searches the external database. If the concrete settings of the simulation model input parameters are found the simulation optimizer does not need to perform the simulation run. If the local and external databases do not contain the required settings, a simulation run is performed and the optimizer saves the calculated objective function value and the concrete settings of the simulation model into the local database. Then it encrypts the data and sends this information to the external database.

The discrete event simulation models - the manufacturing system and logistics model; the assembly line model; the penalty model - were built in Arena simulation software.

3.1 The Manufacturing System and Logistics Model

This discrete event simulation model represents the production of different types of car lights in a whole production system. The complex simulation model describes many processes; for example, logistics in three warehouses, production lines, 28 assembly lines, painting, etc. – see Figure 2.

This discrete event simulation model solves a logistics issue – the transportation of different types of parts from warehouses to assembly lines and the transportation of the final product from the assembly lines to the warehouse. The simulation model was verified and validated according to the real production and logistics system in the company. The model was modified (reducing the number of controls) for the needs of optimization testing. The production plan was fixed for the purpose of testing the optimization methods. The objective of the simulation optimization was to find a suitable number of forklifts for transporting the parts.

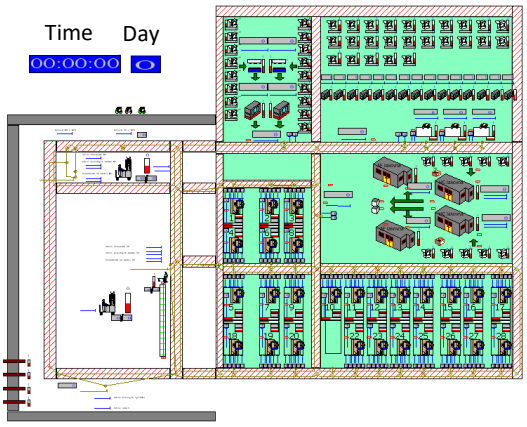

Figure 2. Diagram of the Whole Production System

Decision variables – controls (the input simulation model parameters) are the number of forklifts responsible for:

- Transport of small parts from the warehouse to the production lines and assembly lines
- Transport of large parts from the warehouse to the assembly lines
- Transport of the final product from the assembly lines to the warehouse

The objective function is affected by the sum of the average utilization of all the assembly lines and average transport utilization. The objective function is maximized. The objective function landscape of this model when the number of forklifts for transport of large parts = 14 is shown in Figure 3.

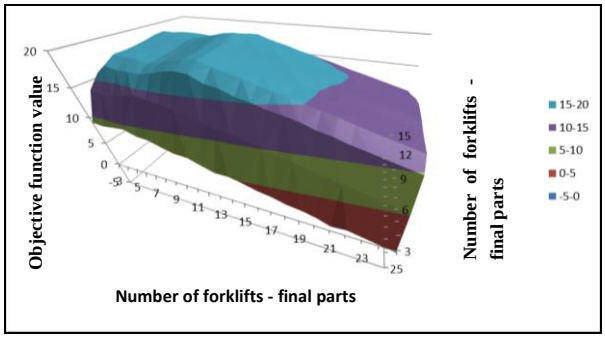

Figure 3. Illustration of the Objective Function of the Manufacturing System and Logistics Model

3.2 The Assembly Line Model

This model represents an assembly line. The products are transported by conveyor belt -Figure 4. The assembly line consists of eleven assembly workplaces. Six of these workplaces have their own machine operator. The rest of the workplaces are automated. A specific scrap rate is defined for each workplace. At the end of the production line there is a sorting process for defective products.

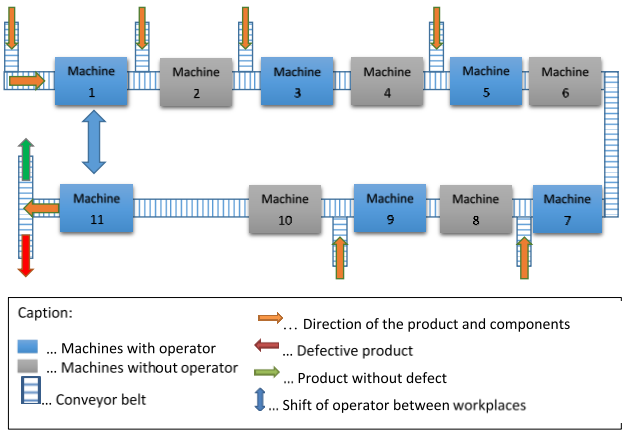

Figure 4. Diagram of the Assembly Line

The decision variables of the simulation model:

- Number of fixtures in the system
- Number of fixtures when the operator has to move from the first workplace to the eleventh workplace to assemble waiting parts on the conveyor belt

The objective function reflects the penalty which is affected by the number of defective products and the pallets in the system. The objective function is maximized. The objective function is shown in Figure 5.

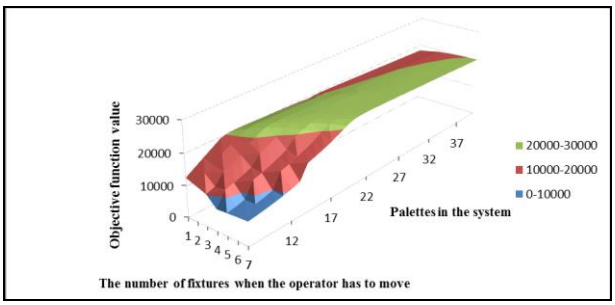

Figure 5. Objective Function of the Assembly Line Model

3.3 *The Penalty Model*

This simulation model represents the production of two types of product. A workshop consists of eight workstations. Each workstation contains a different number of machines. Each product has a specific sequence of manufacturing processes and machining times.

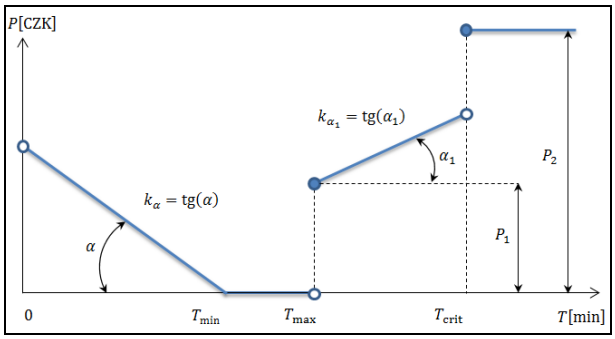

Figure 6. Penalty Function

The product is penalized if the product exceeds the specified production time. A penalty also occurs if the production time value is smaller than the specified constant.

The penalty function is shown inFigure 6 where T denotes production time; $T_{\min}$ denotes required minimum production time; $T_{\max}$ denotes required maximum production time; T_{crit} denotes critical production time; k_α denotes the penalty for early production (slope of the line - constant); k_{α_1} denotes the penalty for exceeding the specified production time (slope of the line - constant); P_1 denotes the penalty for exceeding the specified production time (constant); P_2 denotes the penalty for exceeding the specified critical production time (constant); P denotes the penalty of the product.

Decision variables of the production line simulation model:

- Arrival times of each product in the system

This rule is defined because premature production leads to increasing storage costs – the JIT product.

The objective function is affected by the total time spent by the product in the manufacturing system. The objective function is minimized. The objective function is shown in Figure 7.

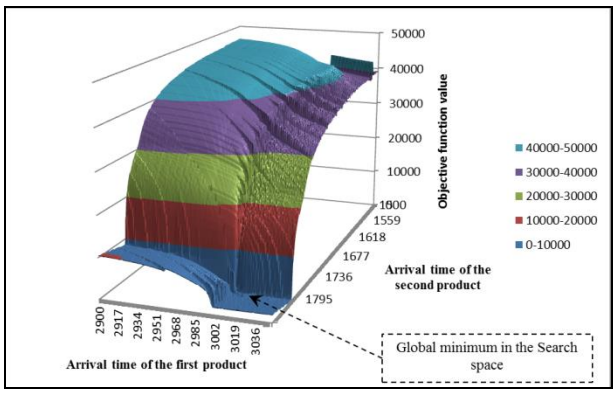

Figure 7. Objective Function of the Penalty Model

3.4 *The Production And Control Stations Model*

Two other discrete event simulation model were built in Tecnomatix Plant simulation software. The Production and Control Stations Model is focused on a production workshop– see Figure 8.

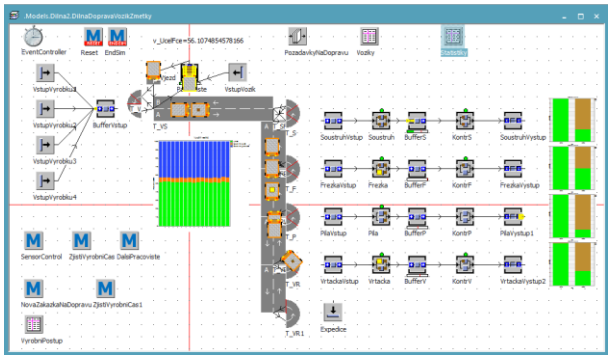

Figure 8. Production And Control Stations Model – Tecnomatix Plant Discrete Event Simulation Model

The simulation model consists of six different workplaces. Workplace number – WN, identifies each workplace.

The product passes through the workplaces respecting the technological progress. Transportation between workplaces uses a forklift truck with a speed of $60\ [m/min]$. The distance between the workplace is 20 metres. Four types of products are processed at the workshop. The first product arrives every 13 minutes, the second product arrives every 5 minutes, the third product arrives every 20 minutes, and the fourth product arrives every 18 minutes at the workshop.

Table 1 gives the sequences of the workplaces which the product passes through. This table also contains the time of processing (and also intervals) at the workplace.

Product	Sequence of Workplaces/Time of Processing [min] - – lower and upper bounds; constant						
	5	1	2	3	4	5	6
1	5	[1,15]	[3,22]	[2,24]	[4,15]	6	
2	5	[1,23]	[3,15]	[2,25]	[3,12]	[4,23]	6
3	5	[1,18]	[4,19]	[3,17]	[4,30]	6	
4	5	[1,25]	[2,26]	[3,25]	[2,15]	[3,25]	6

Table 1. Sequence of the Workplaces

The product is placed in the buffer with a maximum capacity of 15 units before the inspection station after the processing. The time of product inspection is from 20 seconds to 30 seconds (mean - 27 seconds). If the product fails the inspection, the worker must immediately rework the product.

The probability of a defective product (marked as PoDP) and rework time (RT) in minutes using different random distribution at different workplaces (workplace number - WN) are specified for each product.

Decision variables of the simulation model:

- Number of machines at the first, second, third, fourth workplace
- Number of controllers at the first, second, third, fourth workplace
- Number of forklifts

The main goal is to determine the number of machines and controllers at individual workplaces according to the number of lift trucks, machines and controller utilization (maximizing production processes).

The objective function:

$$F(\mathbf{X}) = \frac{\mathrm{NoPP}}{10} + \sum_{WN=1}^{4} (\mathrm{MU} + \mathrm{CU}) \tag{6}$$

Where NoPF denotes the number of processed products; MUdenotes the utilization of the machine; CU denotes the utilization of the controller.

3.5 The Transport Model

The simulation model describes the transport from the warehouse to production lines by tractors. This model illustrates a situation where supply requirements are gradually collected. The tractor with trailers conveys the parts to production lines from the warehouse at regular intervals. The transports are performed according to the requirements of the production lines. Each tractor has defined places to serve and this list of places does not change during the simulation. All supply requirements arise stochastically. The aim of the simulation study is to find the correct sequence of served places at the production lines. The transport time is shorter than the processing time of supplied parts at the production / assembly line. The places for unloading the parts are indexed and the tractor has to pass through these places in ascending order – see Figure 9.

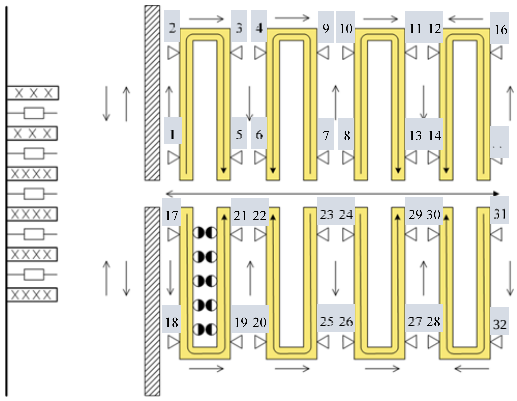

Figure 9. Diagram of the Transportation Path

The simulation model has eight production lines. Each production line has four places for unloading. Four tractors convey the parts from the warehouse. Each tractor pulls a maximum of four trailers. A trailer can convey a maximum of six different boxes with components. Two hundred different parts are conveyed from the warehouse to the production lines. Each tractor can supply from zero to ten places for unloading. The requirement for supplying the parts to the production line follows NegExp (0:20) distribution. Each type of part is defined with a specific probability. Each tractor exits the warehouse every fifteen minutes. The tractors do not leave the warehouse at the same time.

Decision variables of the simulation model:

- Number of places supplied by the tractor (for each tractor)
- Index of first place supplied by tractor (for each tractor)

The objective function reflects the average utilization of tractors with trailers conveying the small and large parts, and the finished product. The objective function also reflects the overall average utilization of the production lines. The average utilization of production lines in the objective function is superior to the average utilization of all types of tractor using the coefficients. The objective function is minimized.

Definitions of the objective functions:

$$F_1(\mathbf{X}) = \sum_{i=1}^{4} DT_i \tag{7}$$

$$F_2(\mathbf{X}) = \frac{\sum_{i=1}^{32} TO_i}{100} \tag{8}$$

$$F_3(\mathbf{X}) = \left(\sum_{i=1}^{200} NT_i\right) * 10000 \tag{9}$$

$$F(\mathbf{X}) = F_1(\mathbf{X}) + F_2(\mathbf{X}) + F_3(\mathbf{X}) \tag{10}$$

WhereDT_i denotes the distance travelled of *i*-th tractor [m]; TO_i denotes the time over the maximum delivery time of the part at the *i*-th place for unloading at the production line [sec]; NT_idenotes the number of transport requests to the *i*-th part to a production line where the part does not have the assigned tractor for the transport [sec]; $F(\mathbf{X})$ denotes the objective function.

4 EVOLUTION STRATEGY

The foundations of the first evolution strategy were laid in the 1960s at the Technical University of Berlin by three students, namely Hans-Paul Schwefel [Schwefel 1995], Ingo Rechenberg and Peter Bienert. Inspired by lectures about biological evolution, they aimed at developing a solution method based on principles of variation and selection. In its first version, a very simple evolution loop without any endogenous parameters was used. [Bäck 2013]

The candidate solution - generated by the optimization method- represents the value of each decision variable of the simulation model. Some optimization algorithms need to access the values of variables hence the element is transformed into a list of values of decision variables (vector of point coordinates in the search space). Decision variables represent the axes in the search space. These axes are indexed from zero to $n-1$:

$$\mathbf{X}[j] = x_j \forall j : j = \{0,1,2,\ldots,n-1\} \tag{11}$$

Where symbols denote: $\mathbf{X}[j]$ denotes the value of the$j$-th decision variable of the simulation model; j denotes the index of the decision variable; $n$denotes the dimension of the search space.

Evolution Strategy generates more than one individual candidate solution. The generated candidate solutions are indexed to distinguish these candidate solutions from each other. Each item can be accessed by the index in the list:

$$\mathbf{X}_i = S[i] \forall i : i = \{0,1,2,\dots,m-1\} \quad (12)$$

Where $\mathbf{X}_i$ is the i-th candidate solution (the individual in the population in the context of Evolution algorithms); S is the list of the generated candidate solutions (population); m is the number of generated candidate solutions (the size of the population).

We implemented the Evolution Strategy algorithm using Steady State Evolution in the simulation optimizer to test different types of selection.[Hynek 2008],[Marik, et al., 2001],[Miranda 2008],[Tvrdik 2004],[Volna 2012]

The optimization algorithm contains the following parameters and functions where: n is the search space dimension; φ is the relative frequency of success; $sum\varphi$ is the sum of relative frequencies of success; $\mathrm{CF}_{F(\mathbf{X})}$ is the comparing function for comparing individuals using their objective function value; A is a list of lower boundaries for each decision variable; B is a list of upper boundaries for each decision variable; j is a j-th decision variable; i is an individual´s index (order in population); σ is a list of standard deviations for each axis (decision variable of the simulation model) of the search space. [Raska 2015]

The standard deviation is affected by Rechenberg 1/5th-rule – line number 25 – see Figure 10. [Schwefel 1995]

ϑ is alist of steps for each decision variable; m is the size of the population; m_{MP} is the number of offspring; q is thenumber of successes (the offspring is better than the parent) to be monitored; k is the number of other contestants per tournament; X_{Pop} is the population of individuals; X_{Arch} is the archive of the offspring.

```
1   begin
2     φ ← ( );
3     σ ← ( );
4     n ← min{Length(A),Length(B)};    //dimension of the search space
5     for j ← 0 to n − 1 do
6       σ ← AddListItem(σ,|(B[j] − A[j])/3|); //initial list of standard deviations
7     X_Pop ← CreatePop(m, A, B);      //initial population
8     while not TerminationCriterion( ) do begin
9       X_Arch ← ( );
10      for i ← 0 to m_MP − 1 do  begin
11        X ← Mutate_ES_n(X_Pop[i mod m], σ, P1, P2, A, B, ϑ);
      //mutation using normal distribution – the offspring of parent
12        if Length(φ) ≥ q then
      //if the relative frequency is too long (the list exceed specified length)
13          φ ← DeleteListItem(φ, 0);    //delete first item in the list - FIFO
14        if CF_F(X)(X, X_Pop[i mod m]) < 0 then
      // the offspring is better than the parent
15          φ ← AddListItem(φ, 1);
      //if the offspring is better than parent add 1 to the end of the list φ else 0
16        else φ ← AddListItem(φ, 0);
17        sumφ ← 0;
18        for i ← 0 to Length(φ) − 1 do
19          sumφ ← sumφ + φ[i];
20        if (sumφ/Length(φ)) < 0.2 then
21          for j ← 0 to n − 1 do begin
22            σ[j] ← σ[j] ∗ 0.82;
23            if σ[j] < ϑ[j]/100 then σ[j] ← ϑ[j] ;
      //increase the standard deviation – not to copy identical gene
24          end
25          else  if (sumφ/Length(φ)) > 0.2 then
26                  for j ← 0 to n − 1 do begin
27                    σ[j] ← σ[j] ∗ 1.22;
28                    if σ[j] > |B[j]−A[j]|/2 then σ[j] ← ϑ[j] ;
      //reduce the standard deviation
29        X_Arch ← AddListItem(X_Arch, X);
30      end;
31      X_Pop ← AppendList(X_Pop, X_Arch);
      //the combined population of parents and offspring
32      AssignFitnessRank(X_Pop);
      // assign fitness (rank) according to objective function value
33      X_Pop ← TournamentSelect_r(X_Pop, m, k);    //tournament selection
34    end;
35    result ← ExtractOptimalSet(X_Pop);  //extract candidate solution(s)
36  end;
```

Figure 10. Evolution Strategy Algorithm - Steady State Evolution[Tvrdik 2004]

TheMutate $_{\mathrm{ES_n}}$function describes the process of individual mutation – line number 11. The detailedalgorithm of the mutation is shown inFigure 11.The function ExtractOptimalSetextracts the best elements from the population. The TerminationCriterion denotes the criterion of stopping the simulation optimization. Theother functions work with the list. The Length function returns the length of the list. The function Delete List Item returns a new list by removing the element at the defined index from the list. The function Add List Item inserts one item at the end of a list. The function App end List adds all the elements of a list to another list.

The next algorithm – see Figure 11 - describes the mutation using parent $\mathbf{X}$ to generate new offspring $\mathbf{X}_{\mathrm{Mut}}$.The mutation uses normal distribution.

```
1   begin
2     Randomize;
3     X_Mut ← X;
4     j ← 0;
5     while j ≤ (Length(X) − 1) do begin
6       if (P1) ≤ Random_u( ) then
7         if ((P2) ≤ Random_u( )) and (j < Length(X) − 2) then begin
8           X_Mut[j] ← X[j + 1];
9           X_Mut[j + 1] ← X[j];
10          X_Mut[j] ← Perturbation_u(X_Mut[j], A[j], B[j]);
11          j ← j + 1;
12        end
13        else X_Mut[j] ← Random_n(X[j], σ[j]);
14      else X_Mut[j] ← Random_n(X[j], σ[j]);
15      X_Mut[j] ← Perturbation_u(X_Mut[j], A[j], B[j]);
16      j ← j + 1;
17    end;
18    result ← X_Mut;
19  end;
```

Figure 11. Mutation of the Parent (using normal distribution) - "Mutate $_{\mathrm{ES\_n}}$"

Parameters in the mutation algorithm: P_1 denotes the probability of mutation; P_2 denotes the probability of swapping neighbouring genes; Random_u denotes the function returning single uniformly distributed random number in interval $[0,1)$. The algorithm uses the "Perturbation" function correcting the individual - mirroring the individual coordinates from the space of the unfeasible solution back to the search space. [Tvrdik 2004]

```
1   begin
2     X_Pert[j] ← X[j];
3     if (A[j] = B[j]) then
4       (X_Pert[j] ← A[j]);
5     while (X[j] < A[j]) or (X[j] > B[j]) do begin
6       if (X[j] < A[j]) then
7         X_Pert[j] ← 2 · A[j] − X[j]
      //mirror the individual coordinate back to search space
8       else if (X[j] > B[j]) then
9           X_Pert[j] ← 2 · B[j] − X[j];
10    X_Pert[j] ← Step(X_Pert[j], A[j], B[j], ϑ[j]);
      //round or truncate the coordinate of the individual in the search space to the nearest neighbour
11    end;
12    result ← X_Pert[j];
13  end;
```

Figure 12. Perturbationfunction[Tvrdik 2004]

Where A denotes the list of lower boundaries for each decision variable; B denotes the list of upper boundaries for each decision variable; $\mathbf{X}_{\mathrm{Pert}}$ denotes a corrected individual and its corrected values of decision variables.

The population is sorted according to the objective function values. The next procedure uses a comparing function for comparing individuals using their objective function value (objective function minimization in this case). The function returns -1 if the first individual is better than the second individual. The function returns 1 if the first individual is worse than the second individual. If the quality of both individuals is the same, the function returns zero.

$$\mathrm{CF}_{F(\mathbf{X})}(\mathbf{X}_1, \mathbf{X}_2) = \begin{cases} -1 & \text{if } F(\mathbf{X}_1) < F(\mathbf{X}_2) \\ 1 & \text{if } F(\mathbf{X}_1) > F(\mathbf{X}_2) \\ 0 & \text{else} \end{cases} \quad (13)$$

Rank-Based Fitness Assignment is a process of assigning a scalar fitness value to each individual in the population according to their order in the population – see Figure 13.

```
1   begin
2     X_Pop ← Sort_a(X_Pop, CF_F(X));
      //sorting of the whole population using comparing function
3     r ← 1;
4     AssignFitnessTo(X_Pop[0], 1) ; //assigning fitness to the first individual
5     for i ← Length(X_Pop) − 1 downto 0 do begin
6       if CF_F(X)(X_Pop[i], X_Pop[i − 1]) < 0 then r ← r + 1;
      //objective function value of the individual X_Pop[i] is better than X_Pop[i − 1]
7       fit ← r;
8       AssignFitnessTo(X_Pop[i], fit);
9     end;
10  end;
```

Figure 13. Rank-Based Fitness Assignment - "Assign Fitness Rank" procedure [Weise 2009]

Parameters in the sorting algorithm: $CF_{F(X)}$ is thecomparing function; r is a candidate solution - individual - order in the population; fit is a fitness value; i is an individual´s index (order in population).

5 ALGORITHMS OF SELECTION

Selection is the process of choosing individuals according to their fitness values from the population and placing them into the mating pool. [Bäck, et al., 2013]

Generally, there are two classes of selection algorithms:

- without replacement(annotated with a subscript $_w$ in the algorithms) - each individual from the population is taken into consideration for reproduction once at most, and therefore will also occur in the mating pool one time at most
- with replacement(annotated with a subscript $_r$) - the mating pool returned by algorithms can contain the same individual multiple times. Like in nature, one individual may thus have multiple offspring.

Another possible classification of selection algorithms is (μ, λ), where μ denotes the number of parents and λ denotes the number of offspring – e.g. (μ, λ) selection strategy is applied to λ offspring while their parents are "forgotten". This selection does not use the information about the parent´s fitness according to the new generation. This strategy relies on anexcess of offspring - the selection uses Darwinian natural selection where λ>μ.[Beyer 2002]

Normally, selection algorithms are used in a variant with replacement. One of the reasons for this is the number of elements to be placed into the mating pool.

The evolutionary algorithms work in exactly the same way – they use a selection algorithm in order to pick the fittest individuals and place them into the mating pool.

The selection algorithms have a major impact on the performance of evolutionary algorithms.[Weise 2009]

5.1 *Random Selection*

Random selection is the simple selection of an individual for reproduction. The offspring are placed into the mating pool. Individuals are selected according to uniform distribution. Each individual has the same probability of selection into the mating pool. Other selection strategies algorithms use a strategy without replacement (annotated with a subscript $_w$ in the algorithms)– the mating pool cannot contain the same individual multiple times.

```
1   begin
2     X_MP ← ( );
3     for i ← 0 to m_MP − 1 do begin
4       j ← ⌊Random_u(Length(X_Pop))⌋;
5       X_MP ← AddListItem(X_MP, X_Pop[j]);
6       X_Pop ← DeleteListItem(X_MP, j);
7     end;
8     Result ← X_MP;
9   end;
```

Figure 14. Random Selection Algorithm - "Random Select$_w$"

5.2 *Truncation Selection*

Truncation (deterministic) selection or threshold selection, returns the best elements from the population. These elements are copied as often as needed until the mating pool size is reached. Population is sorted in ascending order according to the fitness. [Sumathi 2008]

The truncation selection algorithm contains the following parameters: c is the Cut-off value; $Sort_a$ denotes sorting in ascending order considering fitness function; f is thefitness function; $CF_{f(X)}$ is the comparing function for comparing individuals using their fitness function value; X_{Pop} is the population of individuals; X_{MP} is the mating pool; i is an individual´s index (order in population).

```
1   begin
2     X_MP ← ( );
3     c ← min(c, Length( X_Pop));
4     X_Pop ← Sort_a(X_Pop, CF_f(X));
5     for i ← 0 to m_MP − 1 do
6       X_MP ← AddListItem(X_MP, X_Pop[i mod c]);
7     Result ← X_MP;
8   end;
```

Figure 15. Truncation Selection Algorithm - "TruncationSelect[Weise 2009]

5.3 *Ordered selection*

Ordered selection is an approach for reducing the problems of fitness proportionate selection methods. The probability of selection of an individual is proportional to (a power of) its position (rank) in the sorted list of all individuals in the population. The implicit parameter k denotes the selection pressure where:

$$k \in \mathbb{R}^+ \tag{14}$$

The parameter also represents the number of expected offspring of the best individual. A higher value of this parameter leads to higher probability of selection of non-prevalent individuals.

The algorithm of ordered selection is shown in Figure 16 where X_{Pop} denotes a list of individuals who can be selected to the mating pool; m_{MP} denotes the number of selected individuals to the mating pool; k denotes the selection pressure -the number of expected offspring of the best individual, $k \neq m_{MP}$; f denotes thefitness function; q denotes the power value used for ordering; X_{MP} denotes the mating pool.

```
1   begin
2     q ← 1 / (1 − log k / log m_MP);
3     X_MP ← ( );
4     X_Pop ← Sort_a(X_Pop, CF_f(X));
5     for i ← 0 to min{Length(X_Pop), m_MP} − 1 do begin
6       j ← ⌊Random_u()^q * Length(X_Pop)⌋;
7       X_MP ← AddListItem(X_MP, X_Pop[j]);
8       X_Pop ← DeleteListItem(X_Pop, j);
9     end;
10    Result ← X_MP;
11  end;
```

Figure 16. Ordered Selection Algorithm – "Ordered Select$_w$"[Weise 2009]

5.4 Roulette Wheel Selection

Roulette wheel selection is a kind of fitness proportionate selection that has been applied in the original genetic algorithms as introduced by Holland [Holland 1975].

The probability of an individual to enter the mating pool is proportional to its fitness value compared to the sum of the fitness of all individuals. The roulette wheel selection algorithm contains the following parameters: a, b denotes a temporary store for a numerical value; A denotes an array of fitness values; $minf, maxf, sum$ denotes the minimum, maximum, and the sum of the fitness values.

```
1   begin
2     A ⟵ CreateList(Length(X_Pop), 0);
3     minf ⟵ ∞;
4     maxf ⟵ −∞;
5     for i ⟵ 0 to Length(X_Pop) − 1 do begin
6       a ⟵ f(X_Pop[i]);
7       A[i] ⟵ a
8       if a < minf then minf ⟵ a;
9       if a > maxf then maxf ⟵ a;
10    end;
11    if maxf = minf then begin
12      maxf ⟵ maxf + 1;
13      minf ⟵ minf − 1;
14    end;
15    sumNormf ⟵ 0;
16    for i ⟵ 0 to Length(X_Pop) − 1 do begin
17      Normf ⟵ (maxf − A[i]) / (maxf − minf); //fitness maximization: Normf ⟵ (A[i] − minf) / (maxf − minf)
18      A[i] ⟵ Normf;
19      sumNormf ⟵ sumNormf + Normf;
20    end;
21    sum ⟵ 0;
22    for i ⟵ 0 to Length(X_Pop) − 1 do begin
23      Pr ⟵ A[i] / sumNormf;
      //probability of the individual selection
24      sum ⟵ sum + Pr;
25      A[i] ⟵ sum;
      //list containing sum of the probabilities
26    end;
27    for j ⟵ 0 to min{m_MP, Length(X_Pop)} − 1 do begin
      //min function returns the minimum of set
28      a ⟵ Random_u(0, sum);
29      for i ⟵ 0 to Length(X_Pop) − 1 do
30        if (A[i] > a) then break;
      //finding the roulette segment, i.e. the i-th individual index in the population
31      if i = 0 then b ⟵ 0
32      else b ⟵ A[i − 1];
33      b ⟵ A[i] − b;
34      for k ⟵ i + 1 to Length(A) − 1 do
35        A[k] ⟵ A[k] − b;
36      sum ⟵ sum − b;
37      X_MP ⟵ AddListItem(X_MP, X_Pop[i]);
38      X_Pop ⟵ DeleteListItem(X_Pop, i);
39      A ⟵ DeleteListItem(A, i);
40    end;
41    Result ⟵ X_MP;
42  end;
```

Figure 17. Roulette Wheel Algorithm – "Roulette Wheel Select$_w$" [Weise 2009]

6 OPTIMIZATION EXPERIMENTS

We tested all possible solutions of the simulation model so we could specify the value to reach. We also created a database of the simulation experiments. If the simulation optimizer wants to perform the simulation run, the optimizer searches for the simulation experiment in this database. If the simulation experiment is found, the optimizer downloads the simulation experiment and its objective function value.

We tested different settings of the selection strategies parameters – a series of Random, Roulette and Ordered selection. We tested all the selection strategies with or without the replacement of the individual in the population, except for the Truncation selection. Selections are annotated with the letter "R" or "W" in the charts. We tested 23,760 series of the selection strategies on five discrete event simulation models. We replicated these series several times to reduce the random behaviour of the tested selection strategies.

We specified the same conditions which had to be satisfied for each selection strategy, e.g. the same termination criteria. We evaluated the optimization experiments with different settings.

6.1 Termination Criteria

The first termination criterion is VTR (value to reach) – stopping the simulation optimization if the optimum is found. If the optimization algorithm finds a possible solution that its objective function value is within the defined tolerated deviation from the objective function value of the global optimum, the optimization experiment is stopped.

The second termination criterion is that the optimization method could perform a maximum of simulation experiments to find the global optimum in the search space of the discrete event simulation model, which equals $\hat{X}_H$ - the calculated number using the information entropy – see chapter 2.

6.2 Settings of evolution strategy and selection strategies parameters

We defined a step and lower and upper boundaries for the parameters of evolution strategy and selection strategies. If the selection strategy has the same parameters as another selection strategy, we set up both parameters with the same boundaries (same step, lower and upper boundaries).

Parameter	Step	Lower Bound	Upper Bound
m... Size of population	1×n	1×n	6×n
m_{MP}... Number of offspring	1×n	1×n	6×n
q... Number of successes (the offspring is better than the parent) to be monitored	1×n	1×n	6×n
c ... Cut-off value	1×n	1×n	6×n

Table 2. Settings of Evolution Strategy and Selection Strategies Parameters

6.3 Evaluating Criteria

The first criterion is the function whose output is the standardized scalar value $f_1 \in [0,1]$ of not finding the known VTR (value to reach). This value represents the failure of finding the global optimum by the optimization method in a particular series – value minimization. This criterion is expressed by pseudopascal code and shown inFigure 18.

```
1   begin
2     n_Succ ⟵ 0;
3     for i ⟵ 0 to Length(X*) − 1 do
4       if |F(X*[i]) − F(X*)| ≤ ε then
5         n_Succ ⟵ n_Succ + 1 ;
      (*Optimum or acceptable candidate solution
      was found *)
6     result ⟵ (Length(X*) − n_Succ) / Length(X*);
      (*standardization - % share of unsuccessful
      series*)
7     end;
```

Figure 18. Pseudopascal Algorithm of First criterion – Finding the Global Optimum or Suboptimum

This algorithm contains the following parameters: X^* denotes the list of found optima in each optimization experiment in the series; $\mathbf{X}^*$ denotes the global optimum in the search space; ε denotes the tolerated deviation from the value of the objective function value of global optimum; $F(\mathbf{X})$ denotes the objective function value; n_{Succ} denotes the counter of successful finding optimum; f_1 denotes the standardised scalar value.

If the failure is 100[%] the first criterion equals 1, therefore we try to minimize this criterion. Average Method Success of Finding Optimum can be formulated as follows:

$$f_{1AVG} = \left(1 - \frac{\sum_{i=1}^{s} f_{1_i}}{s}\right) \cdot 100[\%] \quad (15)$$

where *i* denotes the index of one series, f_{1_i} denotes the value of the first criterion (Optimization method success – the best value is zero), *s* denotes the number of performed series.

The series were also evaluated regarding specified tolerance between the best optimum (suboptimum) found in the series and the specified parameter ε. The optimization method had to find the candidate solution whose objective function value is nearly the same as the objective function value of the global optimum in the search space. We initially specified $\varepsilon = 0.001$ (the tolerance equals 0.001). We were also interested in allowing a higher tolerated deviation. We calculated the range of the objective function values in the search space:

$$\Delta = |F(\tilde{\mathbf{X}}) - F(\tilde{\mathbf{X}})| \quad (16)$$

Where $F(\tilde{\mathbf{X}})$ denotes the objective function value of the global minimum; $F(\tilde{\mathbf{X}})$ denotes the objective function value of the global maximum. We specified the tolerance 0.1, 0.5 and 1 percent of the objective value range Δ.

The following table shows the objective function value of the global minimum, the global maximum, the dimension of the search space, the number of possible solutions in the search space and the calculated number using the information entropy for the tested discrete event simulation model – see Table 3.

Discrete Event Simulation Model	$F(\mathbf{X})$	$F(\mathbf{X})$	n	$\tilde{X}$	$\tilde{X}_H$
The Manufacturing System and Logistics Model	-11.53367	22.64956	3	8,671	1,454
The Assembly Line Model	7092	22032	2	238	124
The Penalty Model	100.7093	4,6426.34	2	251,001	8,747
The Transport Model	44,502.25	4.26E+07	8	252,000 ,000	74,512
The Production And Control Stations Model	2.149858	57.29105	9	259,200 ,000	74,847

Table 3. Specification of Discrete Event Simulation Models

The following chart shows the average optimization method success of finding the optimum (suboptimum if its objective function value is in the tolerated deviation from the objective function value of the global optimum).

The chart – see Figure 19 - provides us with information about the success of finding the optimum. We can assume high average selection strategies success of finding the optimum (suboptimum) if the dimension of the search space of the discrete event simulation model is lower or the objective function surface is simple - the manufacturing system and logistics model, the assembly line model and the penalty model.

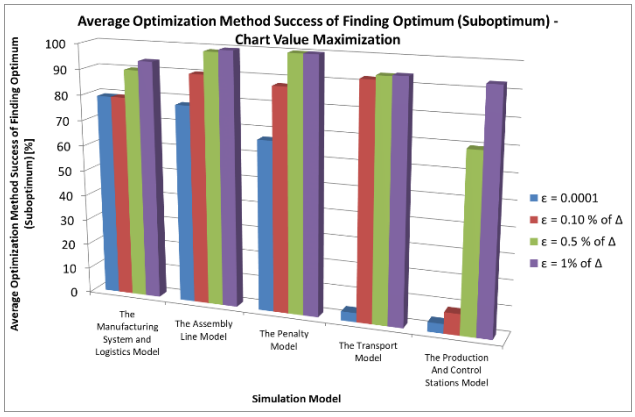

Figure 19. Average Optimization Method Success of Finding Optimum (Suboptimum) of All Tested Selection Strategies – Discrete Event Simulation Models

If the testing function surface is hard – multimodal, planar regions – the optimization failure rate of the optimization selection strategies is high i.e. The Production and Control Stations Model. The next chart contains the absolutely unsuccessful series $f_1 = 1$, i.e. the series does not contain any optimization experiments where the optimum was found – see Figure 20.

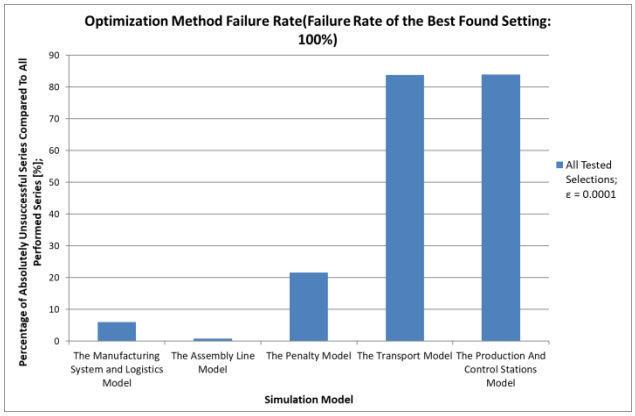

Figure 20. Optimization Method Failure Rate

We tested seven types of selection strategies. The tested selection strategies have a major effect on evolution strategy success – see Figure 21.The chart shows the success of finding the optimum of all the tested selection strategies on the discrete event simulation models. We selected only those series of the optimization where the value of the number of found optimum or suboptimum equals the best found value of the first criterion from all tested series. We found that at least one series had a 100[%] success rate for finding the global optimum from all the performed series of the optimization methods - the first criterion equals 0.

We subsequently counted the number of series with a 100[%] success rate with a specific setting of the optimization method parameter for each simulation model. This number was divided by the total number of series performed with the selected parameter value for a specific model, regardless of the success of a particular series. This value expresses the percentage of the series success of the selection strategy.

We can see that Truncation selection is a very effective strategy for the simulation models where the dimension of the search space is small– see Table 3.If we compare the success of the series of hard termination criterion $\varepsilon = 0.001$ – see Figure 21 - with series success where $\varepsilon = 1\%\ of \Delta$ – see Figure 22.It is obvious that the Random selection is not a very effective method for selecting the individual. The high success of Random selection in the case of the assembly simulation model causes a small number of possible solutions in the search space of this simulation model. The probability of randomly finding the optimum is high.

We can see that each selection strategy has a problem with the complicated objective function landscape – see Figure 21. There are no series with 100[%] success rate of finding the optimum – the transport model and the production and control stations model.

Truncation selection selects the best generated individuals. This approach is fine for a simple objective function landscape. Selection finds the global optimum relatively quickly. Problems can occur if the objective function landscape is multimodal. Losing diversity means a state where all the generated solution candidates are similar to each other – stuck at local optimum. Maintaining a set of diverse solution candidates can prevent being stuck at local optimum – preventing premature convergence.

Preserving diversity is directly linked with maintaining a good balance between exploitation and exploration.[Paenke 2007]

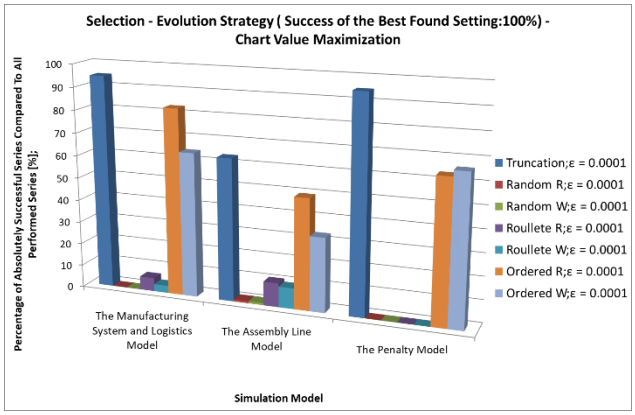

Figure 21. Success of Finding Optimum of the Series Where the Best Found Setting is 100[%];ε = 0.0001

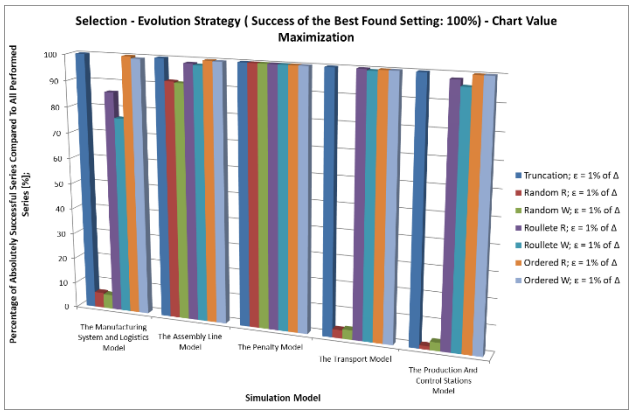

Figure 22. Success of Finding Optimum of the Series Where the Best Found Setting is 100[%];ε = 1% of Δ

We were interested to see how far the objective function values of the best found solutions are from the objective function of the global optimum. We specified a second criterion f_2. This criterion is useful when there is no series which contains any optimum or a solution whose objective function value is within the tolerance of the optimum objective function value (the first criterion $f_1$equals zero in this case). This function evaluates the difference between the objective function value of the best solution found in the series and the optimum objective function value. The list of found optimums considering the objective function value using the comparator function is sorted in ascending order. After that the value of the second criterion is calculated using the formula:

$$f_2 = \left(\frac{F(\mathbf{X}^*) - F(X_{Best})}{F(\mathbf{X}^*) - F(X_{Worst})} \right) \tag{17}$$

where $F(\mathbf{X}^*)$ denotes the objective function value of the global optimum of the search space; $F(X_{Best})$denotes the objective function value of the best solution found in a concrete series; $F(X_{Worst})$denotes the objective function value of the worst found solution (element) of the search space. The output of the function can take the value$f_2 \in [0,1]$.

The average of the second criterion of the absolutely unsuccessful series where $f_1 = 1$is calculated using the formula:

$$f_{2\,AVG} = \left(1 - \frac{\sum_{i=1}^{s} f_{2_i}}{s}\right) \cdot 100[\%] \tag{18}$$

where *i* denotes the index of one series, f_{2_i} denotes the value of the second criterion (optimization method success – the best value is zero), *s* denotes the number of performed series.

The average of the difference between the optimum and the local extreme tested on the discrete event simulation models is shown inFigure 23. The charts contain only series where f_1 = 1 (no optimum was found in the series).

The charts (seeFigure 23) also contain other information. Evolution strategy selections found the global optima very near to the global optimum of the manufacturing system and logistics model, the penalty model, the assembly line model and the transport model. The chart shows that the objective functions values of found global optima were very close to the objective function of the global optimum of the discrete event simulation model. We could say the use of truncation selection and ordered selection should be preferred for the optimization of the simulation models input parameters of tested discrete event simulation models.

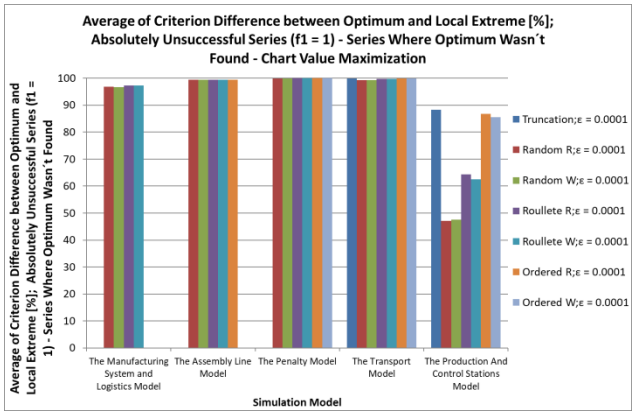

Figure 23. Average of Second Criterion Success: $(1 - f_2) * 100$ [%];

Absolutely Unsuccessful Series ($f_1 = 1$) - Chart Value Maximization; ε = 0.0001

7 CONCLUSION

The goal of the researchwasto test different settings – series - of the selection strategies of the Evolution Strategy on five discrete event simulation models. We evaluated the success of finding the optimum by different selection strategies and we also used a function evaluating the difference between the objective function value of the best solution found in the series and the optimum objective function value.

We deduced that Truncation Selection is a very effective strategy for the simulation models (the manufacturing system and logistics model, the assembly line model and the penalty model) where the dimension of the search space is small and the objective function landscape is not multimodal – see Table 3.Truncation selection selects the best generated individuals. This approach is fine for a simple objective function landscape. The selection method finds the global optimum relatively quickly. A problem can occur if the objective function landscape is multimodal. We confirmed that the success of finding the optimum by the Evolution Strategy selection strategies strongly depends on the objective function landscape. Problems occurred with the finding of the optimum in the higher dimension of the search space of the discrete event simulation model -the production and control stations model.

The selection strategies – "Roulette R – with replacement" and "Roulette W – without replacement" - have almost the same success of finding the optimum. Random selection failed in the case of the higher dimension of the search space.

The chart containing the average of the difference between the optimum and the local extreme shows that the objective functions values of the found global optima are very close to the objective function of the global optimum of the discrete event simulation model.

ACKNOWLEDGMENTS

This paper was created with the subsidy of the project SGS-2018-031"Optimizing sustainable production system parameters" carried out with the support of the Internal Grant Agency of the University of West Bohemia. The paper uses the knowledge and the results from the project CZ.1.07/2.3.00/09.0163 carried out with the support of ESF and the State budget of the Czech Republic.

REFERENCES

[Bäck 2013] Bäck, T. et al. Contemporary Evolution Strategies (1 ed., Vol. XIII). Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2013. ISBN 978-3-642-40136-7

[Beyer 2002] Beyer, H. G. and Schwefel, H. P. Evolution Strategies - A Comprehensive Introduction. Natural Computing, March2002, Vol. 1, No.1, pp. 3-52.ISSN 1567-7818

[Beyer 2017a] Beyer , H. G. and Sendhoff, B. Simplify Your Covariance Matrix Adaptation Evolution Strategy.IEEE Transactions on Evolutionary Computation, March 2017, Vol. PP, No. 99,pp. 1-13. ISSN 1941-0026

[Beyer 2017b] Beyer, H. G. and Sendhoff, B. Toward a Steady-State Analysis of an Evolution Strategy on a Robust Optimization Problem with Noise-Induced Multimodality.IEEE Transactions on Evolutionary Computation, February2017, Vol. 21, No. 4,pp. 629-643. ISSN 1941-0026

[Borda 2011] Borda, M. Fundamentals in Information Theory and Coding. Berlin: Springer Berlin Heidelberg, 2011. ISBN 9783642203473

[Holland 1975] Holland, J. H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor: The University of Michigan Press, 1975. ISBN: 0-4720-8460-7

[Hynek 2008] Hynek, J. Genetic Algorithms and Genetic Programming (in Czech language: Geneticke algoritmy a geneticke programovani). Prague: Grada, 2008. ISBN 978-80-247-2695-3

[Marik 2001] Marik, V. et al. Artificial Intelligence (3).Prague: Academia Praha, 2001. ISBN 80-200-0472-6

[Miranda 2008] Miranda, V. Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems. New Jersey: John Wiley & Sons, 2008. ISBN 9780471457114

[Paenke 2007] Paenke, I. et al. On the influence of Phenotype Plasticity on Genotype Diversity. In: 2007 IEEE Symposium on Foundations of Computational Intelligence. Honolulu, HI, USA, 2007. IEEE, pp. 33-40. ISBN 1-4244-0703-6

[Raska 2015] Raska, P. and Ulrych, Z. Comparison of Optimization Methods Tested On Testing Functions and Discrete Event Simulation Models. International Journal of Simulation and Process Modelling, 2015, Vol 10, No. 3, pp.279-293. ISSN 1740-2123

[Schwefel 1995] Schwefel, H. P. Evolution and Optimum Seeking. New York: John Wiley& Sons, 1995. ISBN 0471571482

[Sumathi 2008] Sumathi, S. et al. Evolutionary Intelligence: An Introduction to Theory and Applications with Matlab. Berlin: Springer-Verlag Berlin Hedilberg, 2008. ISBN 978-3-540-75158-8

[Tvrdik 2004] Tvrdik, J. Evolutionary algorithms - Study Texts (in Czech - Evolucni algoritmy - ucebni texty). Virtual Information Centre for PhD. Students. - University of Ostrava, 2004, Ostrava [online]. [6 February 2012]. Available from http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf.

[Van Rijn 2016] Van Rijn, S. et al. Evolving the structure of Evolution Strategies. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, December 2016.IEEE, pp. 1-8, ISBN 978-1-5090-4240-1

[Volna 2012] Volna, E. Evolutionary Algorithms and Neural Networks (in Czech language: Evolucni algoritmy a neuronove site). University of Ostrava, 2004, Ostrava [online]. [3. May 2016]. Available from http://www1.osu.cz/~volna/Evolucni_algoritmy_a_neuronove_site.pdf

[Weise 2009] Weise, T. E-Book. Global Optimization Algorithms - Theory and Application, 2nd Edition. Thomas Weise, 2009, Germany [online]. [2. February 2011]. Available from http://www.it-weise.de/projects/book.pdf.

CONTACTS:

Ing. Pavel Raska, Ph.D.
University of West Bohemia, Faculty of Mechanical Engineering
Department of Industrial Engineering
Univerzitni 22, Pilsen, 306 14, Czech Republic
Tel.: +420 377 638 415, praska@kpv.zcu.cz
https://www.zcu.cz/about/people/staff.html?osoba=657

Doc. Ing. Zdenek Ulrych, Ph.D.
University of West Bohemia, Faculty of Mechanical Engineering
Department of Industrial Engineering
Univerzitni 22, Pilsen, 306 14, Czech Republic
Tel.: +420 377 638 406, ulrychz@kpv.zcu.cz
https://www.zcu.cz/about/people/staff.html?osoba=17396