EVOLUTION STRATEGY - TESTING DIFFERENT TOURNAMENT SELECTION STRATEGIES

Pavel Raska[(a)], Zdenek Ulrych[(b)]

[(a)] Department of Industrial Engineering - Faculty of Mechanical Engineering, University of West Bohemia, Univerzitni 22, 306 14 Pilsen
[(b)] Department of Industrial Engineering - Faculty of Mechanical Engineering, University of West Bohemia, Univerzitni 22, 306 14 Pilsen

[(a)]praska@kpv.zcu.cz, [(b)] ulrychz@kpv.zcu.cz,

ABSTRACT
The paper deals with testing Evolution Strategy and different tournament selection strategies on testing functions and discrete event simulation model. We developed a simulation optimizer used for simulation optimization, for testing different settings of optimization algorithm parameters and evaluating the success of finding the optimum by the optimization algorithms and other evaluation criteria.

Keywords: evolution strategy selections, testing functions, discrete event simulation model

1. INTRODUCTION

The basic common problem of simulation optimization algorithms is to quickly find the global/local optimum of the objective function (function maximization can be converted to function minimization):

$$\breve{\mathbf{X}} = \operatorname{argmin}_{\mathbf{X}\in\tilde{X}} F(\mathbf{X}) = \{\breve{\mathbf{X}} \in \tilde{X}: F(\breve{\mathbf{X}}) \leq F(\mathbf{X}) \forall \mathbf{X} \in \tilde{X}\} \quad (1)$$

Where symbols denote:

- $\breve{\mathbf{X}}$ … Global minimum of the objective function
- $F(\mathbf{X})$ … Objective function value of candidate solution – the range includes real numbers, i. e. $F(\mathbf{X}) \subseteq \mathbb{R}$. Objective function represents the aim of simulation optimization
- $\tilde{X}$ … Search space

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

$$\tilde{X} = \prod_{j=1}^{n} \tilde{X}_j = \prod_{j=1}^{n} [a_j, b_j], a_j \leq b_j \quad (2)$$

Where symbols denote:

- j… j-th decision variable of the simulation model
- n … Dimension of the search space
- a_j … Lower bound of the interval of j-th decision variable
- b_j … Upper bound of the interval of j-th decision variable

The candidate solution represents the values of each decision variable of the simulation model. Some optimization algorithms (described as pseudo-pascal algorithms) need to access these values hence the element will be transformed into a list of values of decision variables (vector of point coordinate in the search space). Decision variables represents the axes in the search space. These axes are indexed from zero to $n-1$:

$$\mathbf{X}[j] = x_j \forall j: j = \{0,1,2,\dots,n-1\} \quad (3)$$

We have tested different optimization methods (Pseudo gradient – Hill Climbing; Local Search; Tabu Search; Downhill Simplex; Metaheuristic - Simulated Annealing, Evolution Strategy; Differential Evolution and Self Organizing Migrating Algorithm) to find the global optimum of the objective function of the discrete event simulation models. This paper focuses on using Evolution Strategy which is a very general method which can be used for different types of objective functions.

Evolution Strategy generates more than one candidate solution. Each candidate solution represents the individual in the list of the generated candidate solutions – population – in the context of Evolution algorithms.

These candidate solutions are processed in a different way. To distinguish these candidate solutions from each other, the generated candidate solutions are placed into a list. Each item can be accessed by the index in this list:

$$\mathbf{X}_i = S[i] \forall i: i = \{0,1,2,\dots,m-1\} \quad (4)$$

Where symbols denote:

- S … List of candidate solutions
- m … Length of the list S

Most optimization methods are very sensitive to setting their parameters. Hence we tested different settings of Evolution Strategy to reduce some incorrect settings of the optimization methods parameters. We also tested

different selection strategies to find a suitable and effective strategy for the optimization method. Considering the time requirements of testing the optimization method efficiency of finding the optimum of discrete event simulation models we substituted the testing objective function of the simulation models with different testing functions – De Jong´s, Rosenbrock´s, Michalewicz´s and Ackley´s functions.

2. 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, Foussette, and Krause, 2013)

We implemented the Evolution Strategy algorithm using Steady State Evolution in the simulation optimizer to test different types of selection. (Marik, Stepankova, and Lazansky, 2001, Miranda, 2008, Hynek, 2008, Tvrdik, 2004)

The optimization algorithm contains (Raška & Ulrych, 2015) the following parameters and functions:

- n … Search space dimension
- φ … Relative frequency of success
- $sum\varphi$ … Sum of relative frequencies of success
- A … List of lower boundaries for each decision variable (axes of the search space)
- B … List of upper boundaries for each decision variable (axes of the search space)
- j… j-th decision variable
- i… Individual´s index (order in population)
- σ … List of standard deviations for each axis (decision variable of the simulation model) of the search space. The standard deviation is affected by Rechenberg 1/5th-rule – line number 25 (Schwefel, 1995)
- ϑ … List of steps for each axis
- Mutate $_{\mathrm{ES\_n}}$ … Process of individual mutation – line number 11 (algorithm of the mutation is shown in Figure 2. Algorithm parameters are described in the following text)
- m … Size of the population
- m_{MP} … Number of offspring
- q … Number of successes (the offspring is better than the parent) to be monitored
- k … Number of other contestants per tournament
- X_{Pop} … Population of individuals
- X_{Arch} … Archive of offspring
- ExtractOptimalSet … Extracting best elements from the population
- TerminationCriterion … Termination criterion of simulation optimization
- Length … Function providing the length of list
- DeleteListItem … Function returning a new list by removing the element at defined index from the list
- AddListItem … Function inserting one item at the end of a list
- AppendList … Function adding all the elements of a list to another list

```
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, P_1, P_2, 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 1: Evolution Strategy Algorithm - Steady State Evolution

The mutation uses parent $\mathbf{X}$ to generate new offspring $\mathbf{X}_{\mathrm{Mut}}$ - Figure 1: Evolution Strategy Algorithm - Steady State Evolution– line number 11. Mutation uses normal distribution. The mutation is shown in Figure 2.

```
1   begin
2     Randomize;
3     X_Mut ← X;
4     j ← 0;
5     while j ≤ (Length(X) − 1) do begin
6       if (P_1) ≤ Random_u( ) then
7         if ((P_2) ≤ 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 2: Mutation of the Parent (using normal distribution) - "Mutate $_{\mathrm{ES\_n}}$"

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

The mutation algorithm contains the following parameters:

- P_1 … Probability of mutation
- P_2 … Probability of swapping neighbouring genes
- Random_u … Function returning single uniformly distributed random number in interval $[0, 1)$

Algorithm of mutation uses the "Perturbation" function – see Figure 3: "Perturbation" algorithm - using correction of the individual - mirroring the individual coordinates from the space of unfeasible solution back to search space: (Tvrdik, Evolutionary algorithms - Study Texts (in Czech language Evoluční algoritmy - učební texty), 2004)

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

Figure 3: "Perturbation" algorithm

The population is sorted according to the objective function values - Rank-Based Fitness Assignment procedure – see Figure 4: Rank-Based Fitness Assignment - "AssignFitnessRank" (in the context of this paper, fitness is subject to minimization). This sorting algorithm contains the following parameters:

- $\text{CF}_{F(\mathbf{X})}$ … Comparing function for comparing individuals using their objective function value
- r … Individual order
- fit … Fitness value

The procedure uses the process of assigning a scalar fitness value to each solution candidate in the population according to their order in the population.

```
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 4: Rank-Based Fitness Assignment - "AssignFitnessRank" procedure

This procedure sorts all individuals in a population using the comparing function in ascending order. The function compares an individual according to its value of the objective function (objective function minimization):

$$\text{CF}_{F(\mathbf{X})}(\mathbf{X}_1, \mathbf{X}_2) = \begin{cases} -1 & \text{if } F(\mathbf{X}_1) < F(\mathbf{X}_2) \\ 1 & \text{if } F(\mathbf{X}_1) > F(\mathbf{X}_2) \\ 0 & \text{else} \end{cases} \quad (5)$$

3. ALGORITHMS OF THE SELECTION

Selection is the process of choosing individuals according to their fitness values from the population and places them into the mating pool. (Bäck, Foussette, and Krause, 2013)

Generally, there are two classes of selection algorithms:

- with replacement (annotated with a subscript $_r$) - each individual from the population is taken into consideration for reproduction at most once and therefore also will occur in the mating pool one time at most
- without replacement (annotated with a subscript $_w$) - 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) notation where μ denotes the number of parents and λ denotes 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 a new generation. This strategy relies on the excess of offspring - the selection uses Darwinian natural selection where λ> μ. (Beyer and Schwefel, 2002)

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

The selection algorithms have a major impact on the performance of evolutionary algorithms. (Weise, 2009)

3.1. Tournament Selection

Tournament selection proposed by Wetzel (Wetzel, 1983) is one of the popular and effective selection schemes. This type of selection has been analysed by Blickle and Thiele (Blickle and Thiele, 1995), Miller and Goldberg (Sastry and Goldberg, 1996), etc.

The Evolution Strategy optimization algorithm (shown in Figure 1: Evolution Strategy Algorithm - Steady State Evolution) uses Tournament selection - Figure 6. The final population consists of children and parents with good fitness. This strategy supports elitism – the individual with good fitness survives. (Tvrdik, 2010

The next algorithm uses tournament selection with replacement – an individual can compete against itself (its copy). This situation is impossible in reality, but remember, these individuals have the same fitness value therefore the first individual wins and not its copy according to the use of the comparing function. This comparing function selects the first individual if both fitness values are the same (line number 7).

The algorithm creates an empty mating pool (line number 2). Individuals are selected according to their fitness values in the initial stage (line number 3). Individuals are randomly picked from the population (line number 5) and compete with each other in *k* tournaments (line number 7). The winner of these competitions enters the mating

pool (line number 8). Although it is a simple selection strategy, it is a very powerful selection.
The other tournament selection algorithms contain the same following parameters:

- X_{Pop} … List of the individuals to select from
- X_{MP} … Mating pool
- m_{MP} … Number of individuals to be placed into the mating pool
- f … Fitness function
- i … Individual index
- $\text{CF}_{\text{f(X)}}$ … Comparing function for comparing individuals using their fitness function value
- a … Index of the tournament winner
- k … Number of contestants

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

Figure 5: Tournament Selection with replacement - "TournamentSelect $_{\text{r}}$" (Weise, 2009)

The absolute values of the fitness play no role. The only thing that matters is whether or not the fitness of one individual is higher than the fitness of another one, not the fitness difference itself. With rising *k*, the selection pressure increases - individuals with good fitness values create more and more offspring, whereas the chance of worse solution candidates to reproduce decreases. (Weise, 2009)

The tournament selection without replacement (shown in Figure 6: Tournament Selection - "TournamentSelect $_{\text{w1}}$") uses the same principle, but the winner of the tournaments does not participate in other tournaments (is deleted from the population – line number 9).
The second variant of the tournament selection without replacement is identical to the previous algorithm. The difference is that all individuals (their indexes) are selected for one overall round (one list) where the individual cannot be included. The main winner with the lowest fitness value is selected from this list. This process is repeated several times until the mating pool is filled.

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

Figure 6: Tournament Selection - "TournamentSelect $_{\text{w1}}$" (Weise, 2009)

```
1   begin
2     X_MP ← ( );
3     X_Pop ← Sort_a(X_Pop, CF_f(X));
4     for i ← 0 to m_MP − 1 do begin
5       A ← ( );
6       for j ← 1 to min{k,Length(X_Pop)} do begin
7         repeat
8           a ← ⌊Random_u(0,Length(X_Pop))⌋;
9         until Search_u(a, A) < 0;
10        A ← AddListItem(A, a);
11      end;
12        a ← min{A};
13      X_MP ← AddListItem(X_MP, X_Pop[a]);
14    end;
15    Result ← X_MP;
16  end;
```

Figure 7: Tournament Selection - "TournamentSelect $_{\text{w2}}$" (Weise, 2009)

The previous tournament selection algorithms can be called deterministic tournament selection algorithms. The winner of the *k* contestants that take part in each tournament enters the mating pool. In the non-deterministic variant, a probability for the individual selection p is defined. The best individual in the tournament is selected with probability *p*, the second best with probability $p(1-p)$, the third best with probability $p(1-p)^2$ and so on. The *i*-th best individual in a tournament enters the mating pool with probability $p(1-p)^i$. (Weise, 2009)

```
1   begin
2     X_MP ← ();
3     X_Pop ← Sort_a(X_Pop, CF_f(X));
4     for i ← 0 to m_MP − 1 do begin
5       A ← ();
6       for j ← 0 to k − 1 do
7         A ← AddListItem(A, ⌊Random_u(0,Length(X_Pop))⌋);
8       A ← Sort_a (A,CF(a1,a2)≡(f(a1) − f(a2)));
9       for j ← 0 to Length(A) − 1 do
10        if (Random_u() ≤ p) ∨ (j ≥ Length(A) − 1) then begin
11          X_MP ← AddListItem(X_MP, X_Pop[A[j]]);
12          j ← ∞;
13        end;
14    end;
15    Result ← X_MP;
16  end;
```

Figure 8: Tournament Selection - "TournamentSelect $_{r,p}$" (Weise, 2009)

4. SETTINGS OF EVOLUTION STRATEGY AND TOURNAMENT SELECTION PARAMETERS

We tested different settings of the evolution strategy algorithm and tournament selection parameters. We defined a step and lower and upper boundaries for these parameters.

Table 1: Settings of Evolution Strategy And Tournament Selection Parameters

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
k … Number of other contestants per tournament	1×n	1×n	6×n
p … Probability of the individual selection	0.1	0.1	0.6

5. TESTED SIMULATION MODELS

Considering the time required for testing the behaviour of optimization methods, we substitute the testing on the discrete simulation models (and its objective function) by a different testing function to reduce the duration of testing. We tested a different setting of the evolution strategy on four testing functions - De Jong´s, Rosenbrock´s, Michalewicz´s, Ackley´s functions - and discrete event simulation model. (Raska & Ulrych, 2015)

We simulated all feasible solutions of the discrete event simulation model to build a database of simulation experiments to increase the speed of simulation optimization.

5.1. Testing functions

The domain of the function is a defined step for each axis – substitution of the simulation model input parameter (discrete) values of the discrete event simulation model. All testing functions were minimized.

5.2. De Jong´s Function

A convex and unimodal testing function. The function definition: (Pohlheim, 2006)

$$F(\mathbf{X}) = \sum_{j=1}^{n} x_j^2,$$
$$\forall x_j : \left(x_j - \left\lfloor \frac{x_j}{0.01} \right\rfloor \cdot 0.01 = 0 \right) \wedge \left(-30 \le x_j \le 30 \right), \quad (6)$$
$$j = 1:n, n \in [2,10,20,30,44]$$

Where symbols denote:

- $F(\mathbf{X})$… Objective function
- j … Index of control
- n … Dimension of the search space - the dimension of the search space is 10; 20; 30; 40; 50
- x_j … Value of control - testing functions (except Michalewicz function) input parameters values range from -30 (lower boundary) to 30 (higher boundary)

We substitute the testing on the simulation models by testing on the testing function, and we found that the smallest step that can be performed by the optimization methods is 0.01 for each axis in the search space (x_j mod 0.01=0). The input parameters are not continuous.
This resolution represents $8{,}1504\times10^{188}$ possible solutions (combinations of testing function input parameters) in a fifty dimensional search space. To achieve a better idea of the testing functions landscapes the continuous testing functions are shown in the following four figures. De Jong´s continuous function is shown in Figure 9.

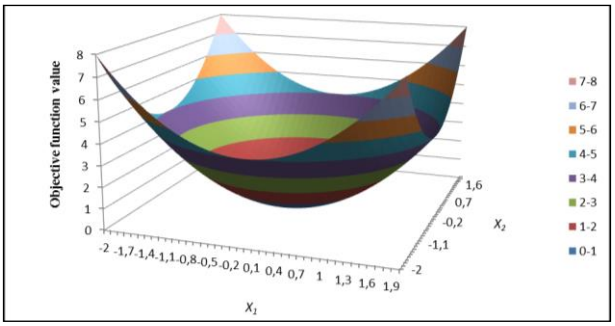

Figure 9: De Jong´s Function

5.3. Rosenbrock´s Function

Rosenbrock´s (Rosenbrock's valley, Rosenbrock's banana) function is a unimodal and non-convex testing function. The function definition: (Pohlheim, 2006)

$$F(\mathbf{X}) = \sum_{j=1}^{n-1} 100 \cdot (x_j^2 - x_{j+1})^2 + (1 - x_j)^2,$$
$$\forall x_j : \left(x_j - \left\lfloor \frac{x_j}{0.01} \right\rfloor \cdot 0.01 = 0 \right) \wedge \left(-30 \le x_j \le 30 \right), \quad (7)$$
$$j = 1:n, n \in [2,10,20,30,44]$$

Rosenbrock´s continuous function is shown in Figure 10.

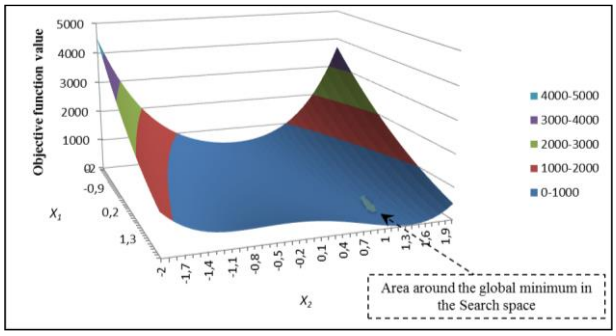

Figure 10: Rosenbrock´s Function

5.4. Michalewicz Function

Michalewicz function is a multimodal test function (n! local optima). The parameter *m* defines the "steepness" of the valleys or edges. Larger *m* leads to a more difficult search. For very large *m* the function behaves like a needle in a haystack (the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum). (Pohlheim, 2006)

$$F(\mathbf{X}) = -\sum_{j=1}^{n} \sin(x_j) \cdot \left(\sin\left(\frac{j \cdot x_j^2}{\pi} \right) \right)^{2 \cdot m},$$

$$\forall x_j : \left(x_j - \left\lfloor \frac{x_j}{0.01} \right\rfloor \cdot 0.01 = 0 \right) \wedge \left(0 \le x_j \le \pi \right), \quad (8)$$

$$j = 1 : n, n \in [2,10,20,30,44]$$

We selected $m = 5$ in our simulation model. The Michalewicz continuous function is shown in Figure 11.

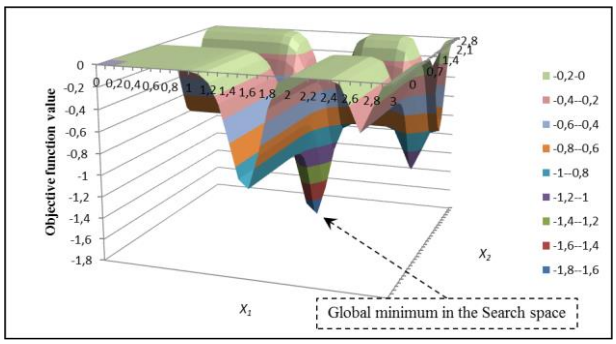

Figure 11: Michalewicz Function

5.5. Ackley´s Functions

Ackley´s function is a multimodal test function. This function is a widely used testing function for premature convergence. (Tvrdik, 2004)

$$F(\mathbf{X}) = -20 \cdot \exp\left(-0.02 \cdot \sqrt{\frac{1}{n} \cdot \sum_{j=1}^{n} x_j^2} \right) - \exp\left(\frac{1}{n} \cdot \sum_{j=1}^{n} \cos 2 \cdot \pi \cdot x_j \right) + 20 + \exp(1), \quad (9)$$

$$\forall x_j : \left(x_j - \left\lfloor \frac{x_j}{0.01} \right\rfloor \cdot 0.01 = 0 \right) \wedge \left(-30 \le x_j \le 30 \right),$$

$$j = 1 : n, n \in [2,10,20,30,44]$$

Ackley´s continuous function is shown in Figure 12.

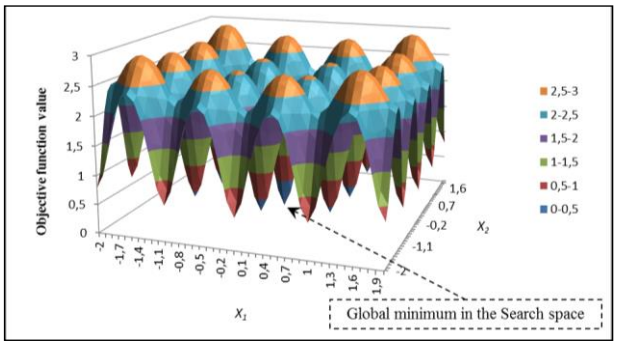

Figure 12: Ackley´s Function

5.6. Discrete event simulation model

The tested discrete event simulation model is focused on a production workshop consisting of the workplaces shown in Table 2: Production workplaces. The simulation model was built in Tecnomatix Plant Simulation software – see Figure 13.

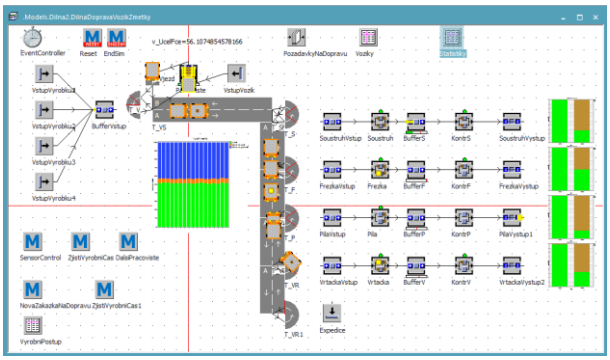

Figure 13: Discrete Event Simulation Model

Table 2: Production workplaces

Workplace Number -WN	Description
1	Sawmill
2	Lathe
3	Milling machine
4	Drill
5	Control station - input
6	Control station - output

The product passes through the following workplaces: $5 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6$. 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.

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

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

Table 3: Sequence of the Workplaces

Product	Sequence of Workplaces/Time of Processing [min]						
	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

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 next table contains 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).

Table 4: Probability of a Defective Product (PoDP) And Rework Time (RT) – The First and Second Product

WN	Product			
	1		2	
	PoDP	RT [min]	PoDP	RT [min]
1	3%	NORM(20,5)	2%	NORM(25,5)
2	5%	NORM(18,5)	4%	NORM(10,5)
3	5%	NORM(20,5)	5%	TRIA(25,30,35)
4	4%	NORM (20,8)	7%	TRIA(14,18,25)

Table 5: Probability of a Defective Product (PoDP) And Rework Time (RT) – The Third and Fourth Product

WN	Product			
	3		4	
	PoDP	RT [min]	PoDP	RT [min]
1	2%	NORM(22,5)	1%	NORM(20,10)
2	4%	NORM(20,5)	3%	NORM(30,10)
3	2%	TRIA(20,25,30)	7%	NORM(20,10)
4	2%	TRIA(20,25,30)	2%	NORM(25,10)

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

The objective function:

$$F(\mathbf{X}) = \frac{\text{NumberOfProcessedProducts}}{10} + \sum_{WN=1}^{4} (\text{MachineUtilization} + \text{ControllerUtilization}) \quad (10)$$

Decision variables of the simulation model (decision variables):

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

6. OPTIMIZATION EXPERIMENTS

We specified the same conditions which had to be satisfied for each optimization method, e.g. the same termination criteria. The optimization method could perform a maximum of 100,000×*n* (parameter *n* denotes the dimension of the search space) simulation experiments to find the global optimum in the search space of the testing function (Tvrdik, Evolutionary algorithms - Study Texts (in Czech language Evoluční algoritmy - učební texty), 2004).

The termination criterion for the discrete event simulation model 10,000×*n* (n=9). Another termination criterion is VTR (value to reach) – stopping the simulation optimization if the optimum is found. 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.

If the optimization method has the same parameters as another optimization method, we set up both parameters with the same boundaries (same step, lower and upper boundaries).

We tested different settings for each evolution strategy selection. The next table (see Table 6) shows the same settings of evolution strategy parameters for each type of tournament selection. Parameter *D* denotes the dimension of the search space. We tested four types of the tournament selection:

- With replacement - "Tournament R" (see Figure 5); "Tournament RP" (see Figure 8)
- Without replacement - "Tournament W1" (see Figure 6); "Tournament W2" (see Figure 7)

If the tournament selection "Tournament R", "Tournament W1" or "Tournament W2" is tested, the parameter - The Number of Contestants - varies from 1×*D* (lower boundary) to 6×*D* (higher boundary) with step 1×*D*. This means that we tested 1,296 different series - different settings - for each selection and also for each testing function or the discrete event simulation model.
If we test the tournament selection "Tournament RP", the parameter - The Probability for The Individual Selection - varies from 0.1 to 0.6 with step 0.1 We have to test 7,776 series for this tournament selection testing for each simulation model.
We tested 233,280 series on the testing functions and 11,664 series on the discrete event simulation model. We replicated these series several times to reduce the random behaviour of the tested optimization method and its selection strategies.

Table 6: Settings of Evolution Strategy Parameters

Method Parameter	Step	Bounds		Number of Series
		Lower Bound	Upper Bound	
Number of Offspring	1×D	1×D	6×D	6*6*6*6 = 1296
Number of Success (the Offspring Is Better Than the Parent) To Be Monitored	1×D	1×D	6×D	
Population Size	1×D	1×D	6×D	
Number of Other Contestants Per Tournament	1×D	1×D	6×D	

We evaluated these optimization experiments with different settings - series. The following charts show the average optimization method success of finding the optimum (suboptimum if the optimum was not found).

The first criterion is the function whose output is the standardized scalar value $\mathfrak{f}_1 \in [0,1]$ 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 value is expressed by pseudopascal code and shown in Figure 14. This algorithm contains the following parameters:

- X^* … List of found optima in each optimization experiment in the series
- $\mathbf{X}^*$ … Global optimum $\mathbf{X}^*$ in the search space
- ε …Tolerated deviation from the value of the objective function value of global optimum
- $F(\mathbf{X})$ … Objective function value
- n_{Succ} …Counter of successful finding optimum
- $\mathfrak{f}_1$ … Standardized scalar value

```
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 14: Pseudopascal Algorithm of First criterion – Finding the Global Optimum or Suboptimum

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:

$$f1\ AVG = \left(1 - \frac{\sum_{i=1}^{s} f_{1_i}}{s}\right) \cdot 100[\%] \qquad (11)$$

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. We initially specified $\varepsilon = 0.001$. 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 (the tolerance equals 0.001).

The following chart 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 testing function is lower or the objective function surface is simple
- If the testing function surface is hard – multimodal, planar regions - the optimization failure rate of the optimization selection strategies is high (the high number of absolutely unsuccessful series $f_1 = 1$, i.e. series does not contain any optimization experiment where the optimum was found)
- Tested tournament selection strategy types have little effect on evolution strategy success
- The success of finding the De Jong´s testing function optimum rapidly decreases if the dimension increases. We obtained high success of finding the optimum on testing function Ackley10.
- Tested selection strategies – "Tournament R" and "Tournament W2" - have almost the same success of finding the optimum

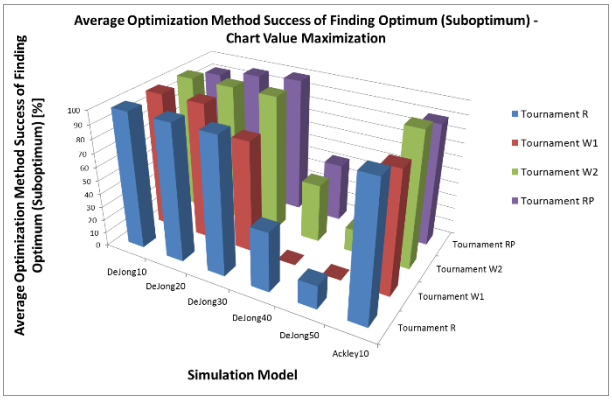

Figure 15: Average Optimization Method Success of Finding Optimum (Suboptimum) – Testing Functions - the Number Denotes the Dimension of the Search Space

- Selection strategies – "Tournament R" and "Tournament W2" – tested on the discrete event simulation model have almost the same success of finding the optimum
- The failure rate (the high number of absolutely unsuccessful series $f_1 = 1$) of finding the optimum of the objective function of the simulation model is almost seventy percent

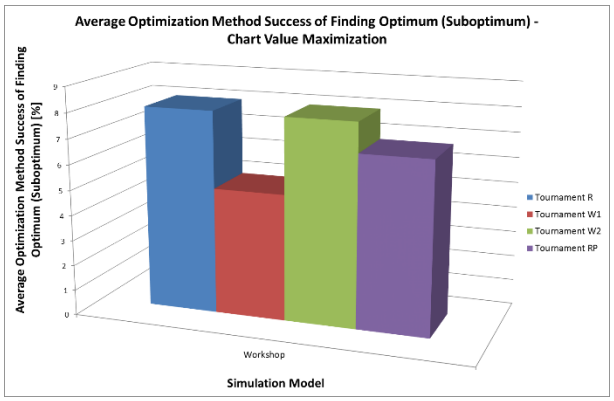

Figure 16: Average Optimization Method Success of Finding Optimum (Suboptimum) – Discrete Event Simulation Model

The second criterion f_2 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 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:

$$f_2 = \left(\frac{F(\mathbf{X}^*) - F(X_{Best})}{F(\mathbf{X}^*) - F(X_{Worst})} \right) \quad (12)$$

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

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

$$f_{2\,AVG} = \left(1 - \frac{\sum_{i=1}^{s} f_{2_i}}{s}\right) \cdot 100[\%] \quad (13)$$

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 testing functions is shown in Figure 17. The charts contain only series where $f_1 = 1$ (no optimum was found in the series).

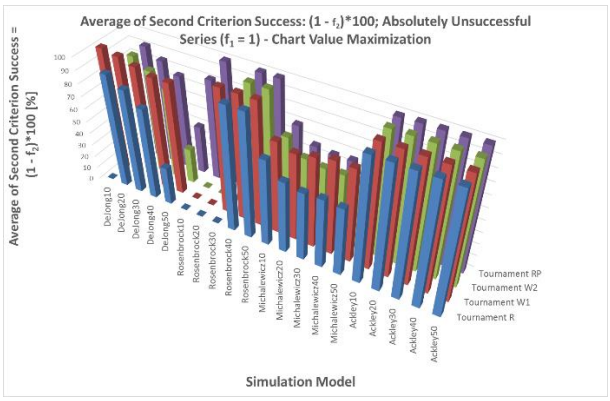

Figure 17: Average of Second Criterion Success: $(1 - f_2) * 100$; Absolutely Unsuccessful Series $(f_1 = 1)$ - Chart Value Maximization – Testing Functions

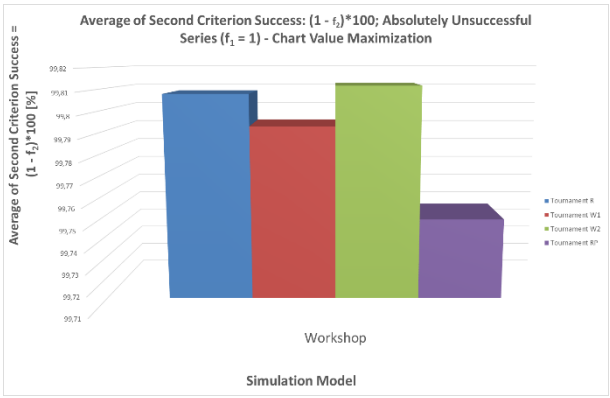

Figure 18: Average of Second Criterion Success: $(1 - f_2) * 100$; Absolutely Unsuccessful Series $(f_1 = 1)$ - Chart Value Maximization – Discrete Event Simulation Model

We can see that each tournament selection type has a problem with the complicated objective function landscape – Michalewicz´s function. The optimization method provides local optima far from the global optimum. The charts (see Figure 17 and Figure 18) also contain other information. Evolution strategy selections found the global optima very near to global optimum of Ackley's function. Tournament selection "Tournament W1" was better (according to the difference between optimum and local extreme) than the other types of selections on average. Averages of the second criterion success are: "Tournament R": 63.7[%]; "Tournament W1": 82.7[%]; "Tournament; W2": 63.5[%]; "Tournament RP": 77.3[%]. Tournament selections "Tournament R" and "Tournament W2" have almost the same success of the second criterion calculated for the testing functions and discrete event simulation model.

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

Objective functions values of found global optima were very close to the objective function of the global optimum of the discrete event simulation model.

7. CONCLUSION

The goal of the research is to test different settings (series) of the tournament selection strategies of the Evolution Strategy on testing functions and a discrete event simulation model. We evaluated the success of finding the optimum by these 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.

The success of finding the optimum by the Evolution Strategy selection strategies strongly depends on the objective function surface. The problems occurred with the finding of the optimum of the Michalewicz's function – flat areas, multimodal function.

Tested selection strategies – "Tournament R" and "Tournament W2" - have almost the same success of finding the optimum. Tournament selection "Tournament W1" was better (according to the difference between the optimum and local extreme) than the other types of selections on average.

ACKNOWLEDGMENTS

This contribution has been prepared within project LO1502 'Development of the Regional Technological Institute' under the auspices of the National Sustainability Programme I of the Ministry of Education of the Czech Republic aimed at supporting research, experimental development and innovation.

REFERENCES

Bäck, T., Foussette, C., and Krause, P., 2013. Contemporary Evolution Strategies,1 ed., Vol. XIII. Berlin, Germany: Springer-Verlag Berlin Heidelberg.

Beyer, H. G., and Schwefel, H. P., 2002. Evolution Strategies - A Comprehensive Introduction. Natural Computing, 1(1), 3-52.

Blickle, T., & Thiele, L. (1995). A mathematical analysis of tournament selection. Proceedings of the Sixth International Conference on Genetic Algorithms, 9-16. Pittsburgh, PA: Morgan Kaufmann Publishers Inc. Available from: http://citeseer.ist.psu.edu/blickle95mathematical.html [accessed March 10, 2015],

Holland, J. H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor: The University of Michigan Press.

Hynek, J., 2008. Genetic Algorithms and Genetic Programming (in Czech language: Genetické algoritmy a genetické programování). Prague: Grada.

Marik, V., Stepankova, O., and Lazansky, J., 2001. Artificial Intelligence (3). Prague: Academia Praha.

Miranda, V., 2008. Fundamentals of Evolution Strategies and Evolutionary Programming. In: El-Hawary, M.E., ed. Modern heuristic optimization techniques. New Jersey: John Wiley & Sons, 43–60.

Pohlheim, H., 2006. GEATbx: Example Functions. Available from: http://www.geatbx.com/docu/fcnindex-01.html#P204_10395 [accessed 20 November 2011]

Raska, P. & Ulrych, Z., 2013. Simulation Optimizer and Optimization Methods Testing On Discrete Event Simulations Models and Testing Functions. ATHENS, GREECE, DIME Universita, pp. 50-59.

Raska, P., and Ulrych, Z., 2015. Comparison of Optimization Methods Efficiency in Different Dimensions of the Search Space of Simulation Models. The 27th European Modeling & Simulation Symposium 2015, pp. 181-190. Bergeggi, Italy: DIME Universita of Genova.

Raška, P., and Ulrych, Z., 2015. Comparison of optimisation methods tested on testing functions and discrete event simulation models. International Journal of Simulation and Process Modelling, Volume 10(Issue 3): pp. 279-293

Sastry, K., and Goldberg, D. E., 1996. Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation. IV, pp. 113-131. Massachusetts Institute of Technology.

Schwefel, H. P., 1995. Evolution and Optimum Seeking. New York, USA: John Wiley& Sons.

Sumathi, S., Hamsapriya, T., and Surekha, P., 2008. Evolutionary Intelligence: An Introduction to Theory and Applications with Matlab (Vol. I). Berlin: Springer-Verlag Berlin Hedilberg.

Tvrdik, J., 2010. Stochastic Algorithms for Global Optimization (in Czech language: Stochastické algoritmy pro globální optimalizaci). Available from: http://www1.osu.cz/~tvrdik/wp-content/uploads/STAGO_10.pdf [accessed 5 January 2014]

Tvrdik, J., 2004. Evolutionary algorithms - Study Texts (in Czech language Evoluční algoritmy - učební texty). Available from: http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf [accessed February 6, 2012]

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

Wetzel, A., 1983. Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial. Pittsburgh, Pennsylvania: University of Pittsburgh.