SIMULATION OPTIMIZER AND OPTIMIZATION METHODS TESTING ON DISCRETE EVENT SIMULATIONS MODELS AND TESTING FUNCTIONS

Pavel Raska[(a)], Zdenek Ulrych[(b)], Petr Horejsi[(c)]

[(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
[(c)] 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, [(c)] tucnak@kpv.zcu.cz

ABSTRACT

The paper deals with testing of selected optimization methods used for optimization of specified objective functions of three discrete event simulation models and four selected testing functions. The developed simulation optimizer uses modified optimization methods which automatically adapt input parameters of discrete event simulation models. Random Search, Hill Climbing, Tabu Search, Local Search, Downhill Simplex, Simulated Annealing, Differential Evolution and Evolution Strategy were modified in such a way that they are applicable for discrete event simulation optimization purposes. The other part of the application is focused on testing the implemented optimization methods. We have proposed some evaluation techniques which express the success of the optimization method in different ways. These techniques use calculated box plot characteristics from the series of optimization experiments.

Keywords: simulation optimization, heuristic optimization methods, discrete event simulation models, testing function

1. INTRODUCTION

Many of today´s industrial companies try to design their own production system as effectively as possible. The problem is that this intention is affected by many factors. We can say the problem is NP-a hard problem in most cases. A possible answer to the problem is using discrete event simulation and simulation optimization. The use of discrete event simulation focuses on the invisible problems in the production system many times and also avoids bad decisions made by the human factor.

The next question is effectively finding a suitable solution to the modelled problem. We can use a simulation optimizer to find an optimal/suboptimal feasible solution respecting the defined model constraints. The basic problem of global optimization can be formulated as follows:

$$\breve{\mathbf{X}} = \arg\min_{\mathbf{X}\in\tilde{X}} F(\mathbf{X}) = \left\{\breve{\mathbf{X}} \in \tilde{X} : F(\breve{\mathbf{X}}) \leq F(\mathbf{X}) \forall \mathbf{X} \in \tilde{X}\right\} \quad (1)$$

where $\breve{\mathbf{X}}$ denotes the global minimum of the objective function; $F(\mathbf{X})$ denotes the objective function value of the candidate solution – the range includes real numbers; $\tilde{X}$ denotes the Search space. This optimal solution is represented by the best configuration (input parameters values) of the simulation model.

Current simulation software (Arena, Witness, PlantSimulation etc.) uses its own simulation optimizers. These integrated optimization modules are black-boxes but many of them use similar optimization methods. We have tested the following optimization methods: Random Search, Hill Climbing, Tabu Search, Local Search, Downhill Simplex, Simulated Annealing, Differential Evolution and Evolution Strategy. These methods were modified in such a way that they are applicable for discrete event simulation optimization purposes. The goal of our research is to compare some of these widely used optimization methods. Hence we have designed our own simulation optimizer. We have to say that it is not possible to implement exactly the same optimization methods which are used in these simulation optimizers.

Another reason for testing the optimization methods and designing our own simulation optimizer was that our department focuses on modelling and optimizing production and non-production processes in industrial companies (Kopecek, 2012; Votava, Ulrych, Edl, Korecky and Trkovsky, 2008). Some projects have to be solved with difficulty without the use of special simulation optimization tools because of the large complexity of the discrete event simulation model. The problem is that some integrated simulation optimizers cannot affect all the parameter types of the designed simulation model.

2. SELECTED OPTIMIZATION METHODS

We have transformed some of the selected optimization methods to use the principle of evolutionary algorithms. These optimization methods generate a whole population (instead of one possible solution) in order to avoid getting stuck on a local optimum. Previous testing of optimization methods confirms that generating one solution leads to premature convergence in most cases (depending on objective function type). Different variants of selected optimization methods obtained from a literature review were united into the algorithm. The user can combine different variants of optimization methods by setting the optimization method parameters.

2.1. Random Search

A new candidate solution is generated in the search space with uniform distribution (Monte Carlo method). This method is suitable for cases where the user has no information about the objective function type. The user is able to perform a number of simulation experiments.

2.2. Downhill Simplex

This method uses a set of n + 1 linearly independent candidate solutions (n denotes search space dimension) - Simplex. The method uses four basic phases – Reflection, Expansion, Contraction and Reduction. (Tvrdík 2004; Weise 2009)

2.3. Stochastic Hill Climbing

Candidate solutions are generated (populated) in the neighbourhood of the best candidate solution from the previous population. Generating new possible solutions is performed by mutation. This method belongs to the family of local search methods.

2.4. Stochastic Tabu Search

The newly generated candidate solution is an element of the Tabu List during the optimization process. This candidate solution cannot be visited again if the aspiration criterion is not satisfied (this feature prevents the method from becoming stuck at a local optimum). The method uses the FIFO method of removing the candidate solution from the Tabu List. The user can set whether the new candidate solution is generated using mutation of the best candidate solution from the previous population or the new solution is generated using mutation of the best found candidate solution. (Monticelli, Romero and Asada 2008; Weise 2009)

2.5. Stochastic Simulated Annealing

A candidate solution is generated in the neighbourhood of the candidate solution from the previous iteration. This generating could be performed through the mutation of a randomly selected gene or through the mutation of all genes. Acceptance of the worse candidate solution depends on the temperature. Temperature is reduced if the random number is smaller than the acceptance probability or the temperature is reduced if and only if a worse candidate solution is generated. If the temperature falls below the specified minimum temperature, temperature is set to the initial temperature. (Monticelli, Romero and Asada 2008; Weise 2009)

2.6. Stochastic Local Search

A candidate solution is generated in the neighbourhood of the best candidate solution.

2.7. Evolution Strategy

This optimization method uses Steady State Evolution – population consists of children and parents with good fitness. A candidate solution (child) is generated in the neighbourhood of the candidate solution (parent) and it is based on the Rechenberg 1/5th-rule. The population is sorted according to the objective values (Rank-Based Fitness Assignment). The optimization method uses Tournament selection. (Koblasa, Manlig and Vavruska 2013; Miranda 2008; Tvrdík 2004)

2.8. Differential Evolution

Selection is carried out between the parent and its offspring. The offspring is created through a crossover between the parent and the new candidate solution (individual) which was created through the mutation of four selected individuals and the best one selected from the population – BEST method. The optimization method uses General Evolution and the Ali and Törn adaptive rule. The user can define the probability of a crossover between the new candidate solution and the parent. (Tvrdík 2004; Wong, Dong, 2008)

3. DEVELOPED APPLICATION

We have developed our own simulation optimization application which addresses the problems listed in the first chapter. The application contains seven different global optimization methods. This application contains two modules. The first module is a simulation optimizer which enables optimization of developed simulation models in ARENA or PlantSimulation simulation software. The objective function of the models is specified within the discrete simulation models. The user can also test a specified objective function without the need of creating the simulation model – Figure 1.

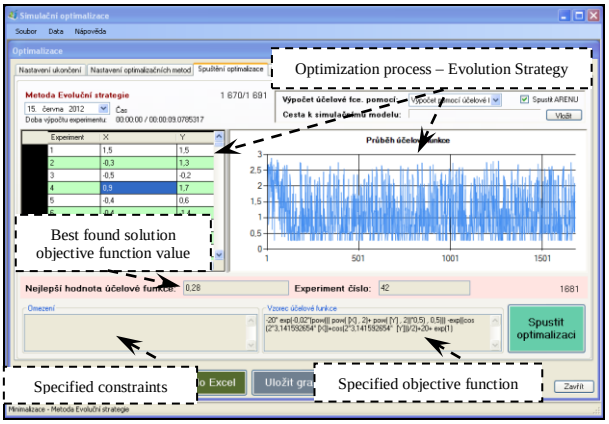

Figure 1: Graphical user interface of simulation optimizer - first module

The application was created in Visual Basic 2010. This programming language was used for connection to ARENA simulation software and Microsoft Access database. The data from simulation experiments results and settings are stored in this database. The file contains information about:

1. Controls – identification, names, low and high boundaries, type (discrete vs. continuous), initial values of controls and comments.
2. Constraints – specification of the constraint function through using the mathematical operator buttons and the list of controls. User can validate the built expression.
3. Objective function - specification of the objective function without the need of a simulation software tool. Objective function is composed of mathematical operators and selected controls from the list of all controls. User can validate the built expression.
4. Optimization experiment setting – minimization vs. maximization of objective function, Termination criterion (Value to reach, number of simulation experiments, specified time, sub-optimum improvement ratio etc.), parameters settings of selected optimization method, low and high boundaries of selected optimization method parameters, number of replications, creation of a knowledge base of a simulation model, etc.

The second module is designed for testing the behaviour of the implemented optimization method in terms of setting the parameters for the optimization method. The user can specify the range of optimization method parameters. After finishing the number of optimization experiments replications (series of concrete optimization method setting) the data are exported to MS Excel workbook.

We have also developed an application which enables 3D visualization of simulation experiments when there are two controls and one objective function. Simulation experiments are represented by the points in the 3D chart of the objective function. The objective function surface is generated from the data obtained from simulation experiments. Another possibility is to generate a whole 3D chart from the data obtained from the simulation runs of all possible settings of the simulation model input parameters – complete search space.

4. DISCRETE EVENT SIMULATION MODELS AND OBJECTIVE FUNCTIONS

The testing of optimization methods which search for global optima was applied to three discrete event simulation models. These models reflect real production systems of industrial companies. Discrete event simulation models were built in Arena simulation software. We specified different objective functions considering the simulated system. All possible solutions and their objective function values were mapped to find the global optimum in the search space.

4.1. The Manufacturing System and Logistics

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. The objective function is affected by the sum of the average utilization of all assembly lines and average transport utilization. The objective function is maximized. Controls 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 and the transport of the final product from the assembly lines to the warehouse. The objective function landscape of this model when the number of forklifts for transport of large parts = 14 is shown in Figure 2.

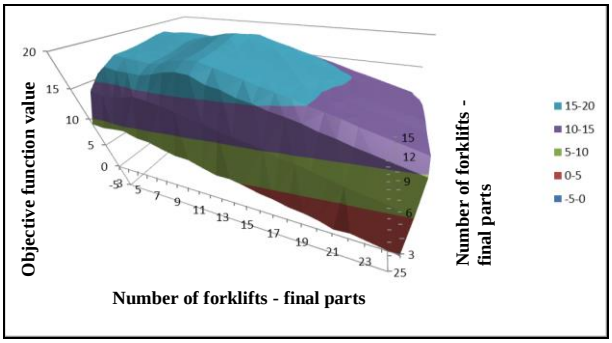

Figure 2: Objective Function - The Manufacturing System And Logistics Discrete Event Simulation Model - Number of Forklifts for Large Parts = 14

4.2. The Penalty

This simulation model represents a production line which consists of eight workstations. Each workstation contains a different number of machines. Each product has a specific sequence of manufacturing processes and machining times. 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 in Figure 3 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.

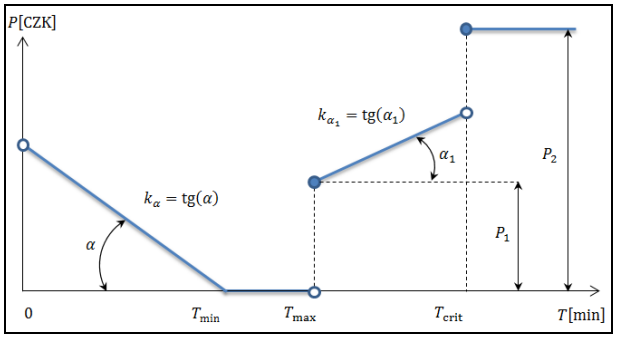

Figure 3: Penalty Function

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. Controls of the production line simulation model are the arrival times of each product in the system. The objective function is shown in Figure 4.

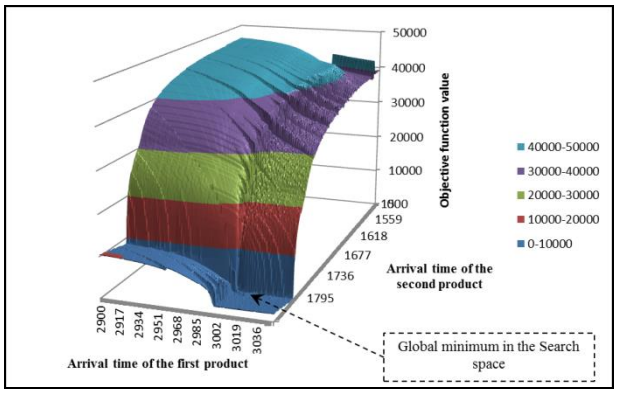

Figure 4: Objective Function – The Penalty Discrete Event Simulation Model

4.3. The Assembly Line

This model represents an assembly line. Products are conveyed by conveyor belt. 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 is a sorting process for defective products. The objective function reflects the penalty which is affected by the number of defective products and the palettes in the system. The objective function is maximized. The objective function is shown in Figure 5. The input simulation model parameters (controls) are the numbers of fixtures in the system and the 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.

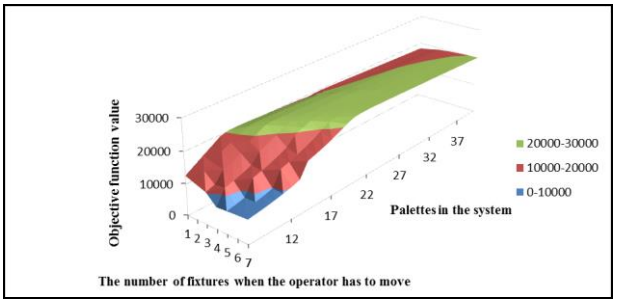

Figure 5: Objective function - The Assembly Line discrete event simulation model

5. TESTING FUNCTIONS

We also tested implemented optimization methods on four standard testing functions. All testing functions are minimized.

5.1. De Jong´s Function

It is a continuous, convex and unimodal testing function. The function definition:

$$F(\mathbf{X}) = \sum_{j=1}^{n} x_j^2 \tag{2}$$

where $F(\mathbf{X})$ denotes the objective function; j denotes index of control; n denotes the dimension of the search space; x_j denotes the value of control. The objective function is shown in Figure 6.

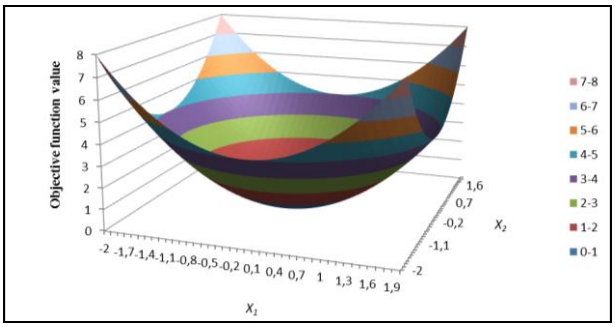

Figure 6: Objective Function - De Jong´s Function

5.2. Rosenbrock´s Function

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

$$F(\mathbf{X}) = \sum_{j=1}^{n-1} 100 \cdot (x_j^2 - x_{j+1})^2 + (1 - x_j)^2 \tag{3}$$

The objective function is shown in Figure 7.

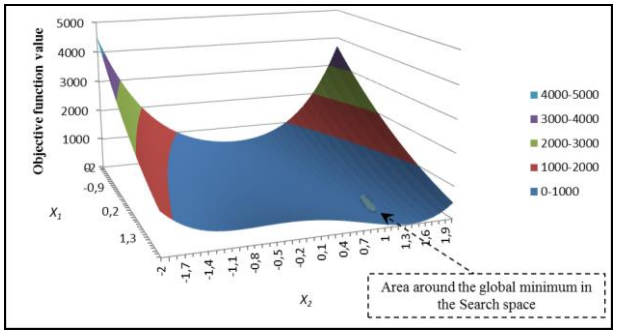

Figure 7: Objective Function - Rosenbrock´s Function

5.3. Michalewicz´s Function

Michalewicz´s 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)

$$F(\mathbf{X}) = -\sum_{j=1}^{n} \sin(x_j) \cdot \left(\sin\left(\frac{j \cdot x_j^2}{\pi} \right) \right)^{2 \cdot m} \quad (4)$$

$$j = 1:n, 0 \le x_j \le \pi \quad (5)$$

We selected $m = 5$ in our simulation model. The objective function is shown in Figure 8.

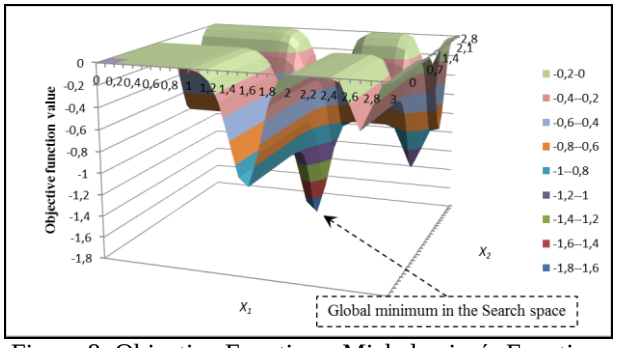

Figure 8: Objective Function - Michalewicz´s Function

5.4. Ackley´s Functions

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

$$F(\mathbf{X}) = -20 \cdot \exp\left(-0.02 \cdot \sqrt{\frac{1}{n} \cdot \sum_{j=1}^{n} x_j^2} \right) - \exp\left(\frac{1}{n} \cdot \sum_{j=1}^{n} \cos 2 \cdot \pi \cdot x_j \right) + 20 + \exp(1) \quad (6)$$

$$j = 1:n, -30 \le x_j \le 30 \quad (7)$$

The objective function is shown in Figure 9.

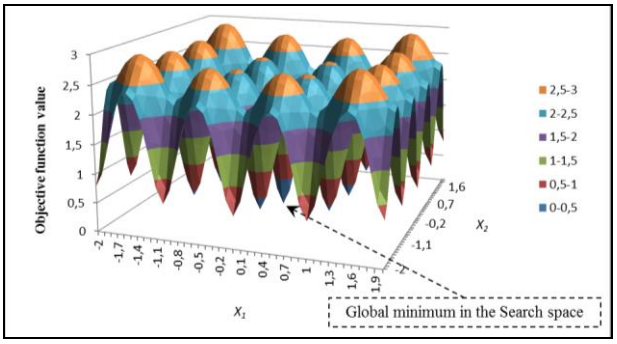

Figure 9: Objective Function - Ackley´s Function

6. EVALUATION METHOD

Simulation experiments results are saved to a database file during simulation experiments if the user uses a simulation optimizer. Simulation experiments results are visualized in the objective function chart and stored in the table placed in the application. The graphical user interface of the first module is shown in Figure 1.

If the second module is used the simulation experiments data are exported to MS Excel workbook after finishing the series (series - replications of optimization experiments with concrete optimization method setting). Excel was selected because of its wide usage

Considering the number of simulation experiments we can divide the number of simulation experiments – Figure 10:

1. Simulation experiment – simulation run of simulation model.
2. Optimization experiment – performed with concrete optimization method setting to find optimum of objective function.
3. Series – replication of optimization experiments with concrete optimization method setting.

The second module focuses on testing the behaviour of the implemented optimization method in terms of setting the parameters for the optimization method. The user can set up the parameters of a selected optimization method, low and high boundaries of the selected optimization method parameters, number of replications, and export the objective function chart to image – Figure 11.

The same conditions had to be satisfied for each optimization method, e.g. the same termination criteria, the same search space. If the optimization method has the same parameters as another optimization method, we set up both parameters with the same boundaries (same step, low and high boundaries).

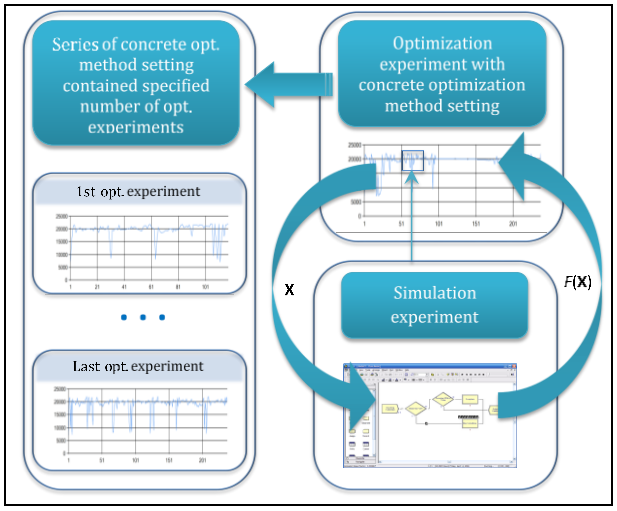

Figure 10: The Number of Simulation Experiments

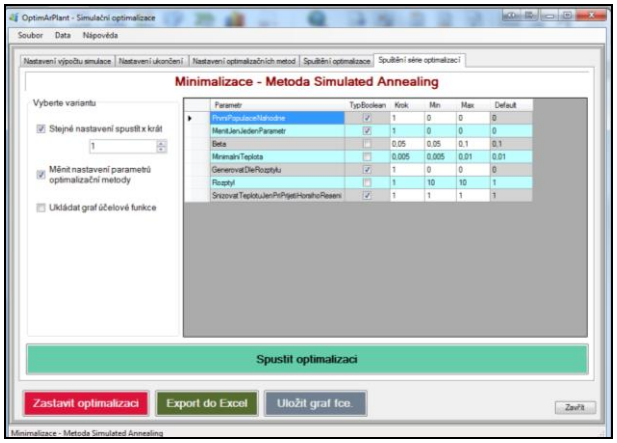

Figure 11: GUI of the Second Module

Box plot characteristics (the smallest observation – sample minimum Q_1, lower quartile Q_2, median Q_3, upper quartile Q_4, and largest observation - sample maximum Q_5) are calculated for each performed setting of the optimization method parameters – Figure 12.

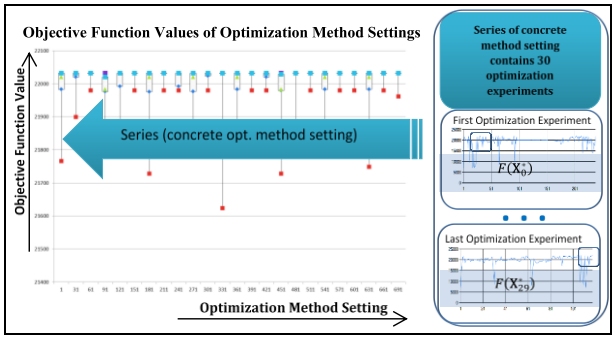

Figure 12: Example of Results from Simulation Optimization Experiments Provided by Evolution Strategy Displayed in Box Plot Chart – The Assembly Line Simulation Model

$F(\mathbf{X}_0^*)$ denotes the found optimum (local in this case). These characteristics are visualized in the box plot chart – Figure 12.

Three box plot charts are generated - Best objective function value, Range of provided function objective values during the simulation experiments, and Number of experiments required to find global (local) optimum. Visualization can help the user to find a suitable setting of optimization method more quickly.

Due to the large volume of data (over 4 billion simulation experiments) we have to propose evaluation techniques (criteria) which express the failure of the optimization method in different ways. Each criterion value is between [0, 1]. If the failure is 100[%] the criterion equals 1 therefore we try to minimize all specified criteria. We implemented the graphical user interface to MS Excel workbook which enables the user to set up the weights of each criterion and other parameters of the evaluation. These parameters are automatically loaded from the simulation experiments results. We used the VBA for MS Excel.

6.1. Optimization Method Success

The first criterion f_1 is the value of not finding the known VTR (value to reach). This value is expressed by:

$$f_1 = \frac{s - n_{succ}}{s} \tag{8}$$

where s denotes the number of performed series, n_{succ} denotes the series where the VTR was found. Simulation runs of all possible settings of simulation model input parameters were performed. This means that we have evaluated all possible solutions of the search space hence we can determine the global optimum (VTR) in the search space. Average Method Success of Finding Optimum can be formulated as follows:

$$f_{avg} = \left(1 - \frac{\sum_{i=1}^{s} f_{1_i}}{s}\right) \cdot 100[\%] \tag{9}$$

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 average optimization method success of finding the optimum of testing functions is shown in Figure 13.

We can see that the Evolution Strategy and Simulated Annealing are successful optimization methods. Random Search also achieves good results. It was affected by doing many simulation experiments by this method. The probability of finding the optimum increases with a high number of simulation experiments. This strategy is simply random and if the search space is huge (NP-hard) we can say it is lucky to find the optimum. This method is usable when the user has no information about the objective function type. We have to evaluate each possible solution in the search space to obtain the optimum hence the search space cannot be too huge.

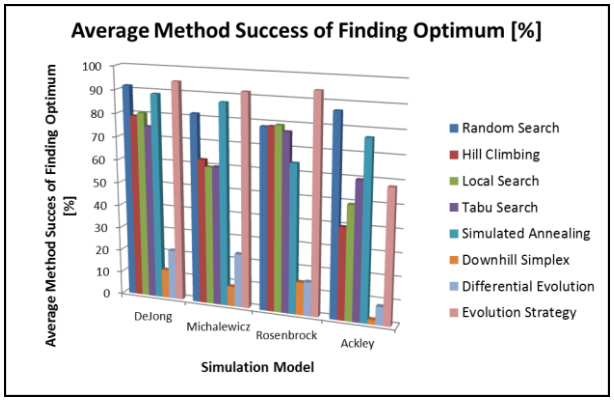

Figure 13: Average Optimization Method Success – Simulation Optimization Results of Testing Functions

Average optimization method success of finding the optimum of discrete event simulation models is shown in Figure 14. We can say that Simulated Annealing and Evolution Strategy are quite successful optimization methods again. Random Search was not successful in the case of the Penalty model because of the larger search space. The Penalty discrete event simulation model has a complicated objective function landscape. The area around the optimum is straight and the method could not obtain information about rising or decreasing the objective function terrain.

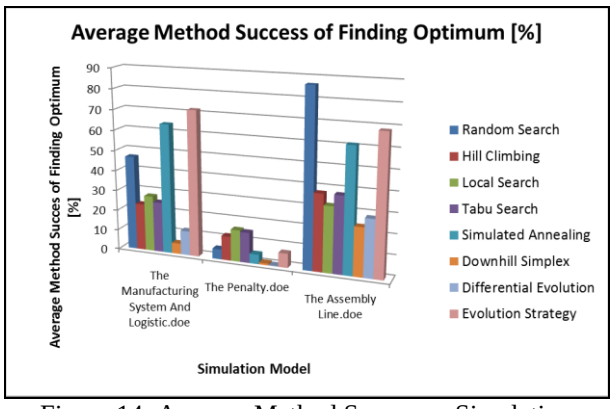

Figure 14: Average Method Success – Simulation Optimization Results of Discrete Event Simulation Models

Previous charts express the average success of optimization methods of all optimization methods settings. These charts also contain bad settings therefore we separated the bad series from the good series. The next chart contains the filtered series with the best found first criterion value only (in this case $f_1 = 0$ so the optimum was found in each optimization experiment). The percentage of absolutely successful series compared to all performed series is shown in Figure 15. It is obvious that the favourite, Evolution Strategy, has problems with the multimodal Ackley function. The success of this method was affected by the number of individuals randomly chosen from the population for the tournament – exploration vs. exploitation of the search space.

The first approach is to generate other new solutions which have not been investigated before - exploration. Since computers have only limited memory, the already evaluated solution candidates usually have to be discarded in order to accommodate new ones. Exploration is a metaphor for the procedure which allows search operations to find new and maybe better solution structures. Exploitation, on the other hand, is the process of improving and combining the traits of the currently known solutions, as done by the crossover operator in evolutionary algorithms, for instance. Exploitation operations often incorporate small changes into already tested individuals leading to new, very similar solution candidates or try to merge building blocks of different, promising individuals. They usually have the disadvantage that other, possibly better, solutions located in distant areas of the problem space will not be discovered. (Michalewicz 2004)

The behaviour of Hill Climbing, Local Search and Tabu Search is similar considering the similar pseudo gradient principle.

Substandard results were achieved with the Downhill Simplex method. This optimization method works by calculating the points of the centroid (center of gravity of the simplex). We have to modify this optimization method in such a way that it is applicable for discrete event simulation optimization purposes where the step in the search space is defined. We use the rounding of coordinates of the vector (new calculated point) to the nearest feasible coordinates in the search space and this leads to deviation from the original direction. We performed other simulation experiments with smaller steps and the success of finding the optimum was higher than before. This problem can be solved by using a calculation with the original points and the objective function value will be calculated by the approximations of the objective value of the nearest feasible points in the search.

Differential Evolution uses the elitism strategy in our case. This leads to copying of identical individuals which suppresses the diversity of new promising individuals. Random Search looks successful, but there were only two possible settings – generating the same individual possibility. This evaluation can be modified by using the coefficient which recalculates the value of success depending on the number of performed series. The termination criterion was the number of possible

solutions in the search space when there is little search space. This led to increasing the probability of success of this optimization method.

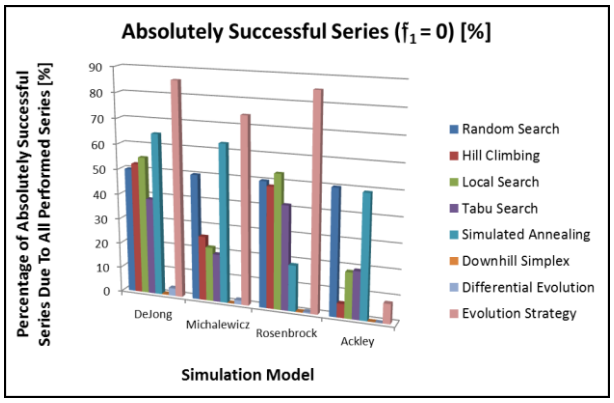

Figure 15: Percentage of Absolutely Successful Series Compared To All Performed Series - Testing Functions

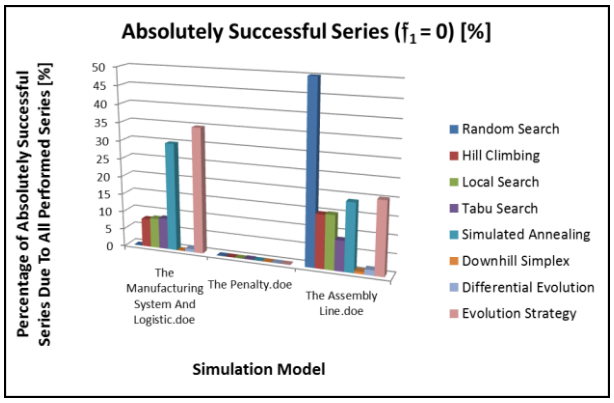

Figure 16: Percentage of Absolutely Successful Series Considering All Performed Series - Discrete Event Simulation Models

6.2. The Difference between Optimum and Local Extreme

The second criterion f_2 is useful when there is no series which contains any optimum or the solution whose objective function value is within the tolerance of optimum objective function value. The first criterion f_1 equals zero in this case. The function where the output of the function can take value $f_2 \in [0,1]$. 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 effort is to minimize f_2. 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:

$$f_2 = \frac{\left|F(\mathbf{X}^*) - F(X_{\text{Best}})\right|}{\left|F(\mathbf{X}^*) - F(X_{\text{Worst}})\right|} \qquad (10)$$

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 concrete series; $F(X_{\text{Worst}})$ denotes objective function value of the worst solution (element) of the search space.

The difference between the optimum and the local extreme is shown in Figure 17 (testing functions) and Figure 18 (discrete event simulation models). The charts contain only series where the f_1 = 0 (no optimum was found in the series). The average of second criterion f_2 is shown for each optimization method – these values express the failure of the optimization method. Output of function can take value $f_2 \in [0,1]$.

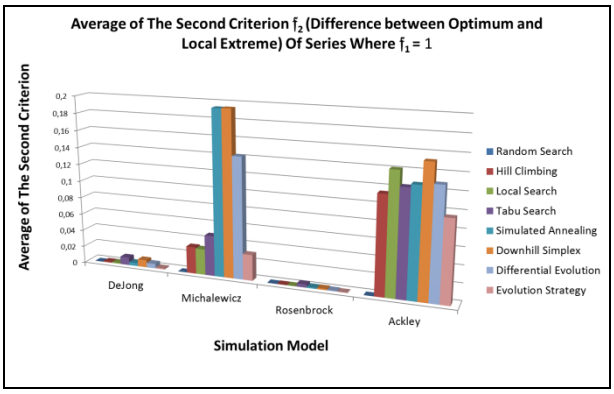

Figure 17: Average of the Second Criterion f_2 - Difference between Optimum and Local Extreme - Testing Functions

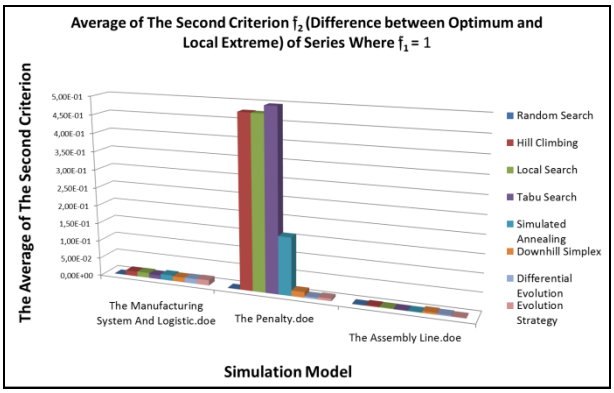

Figure 18: Average of the Second Criterion f_2 - Difference between Optimum and Local Extreme - Discrete Event Simulation Models

6.3. The Distances of Quartiles

Third criterion f_3 expresses the distance between quartiles of a concrete series. Weights are used for evaluation purposes. These weights penalize the solutions) placed in quartiles. Values of the weights were defined based on the results of the simulation experiments. The user can define the weight value. The sum of weights equals one. The third criterion when the objective function is minimized can be formulated as follows:

$$f_3 = \frac{|Q_1 - F(\mathbf{X}^*)| + w_{4f_3}|Q_1 - Q_2| + w_{3f_3}|Q_2 - Q_3| + w_{2f_3}|Q_3 - Q_4| + w_{1f_3}|Q_4 - Q_5|}{|F(\mathbf{X}^*) - F(X_{\text{Worst}})|} \quad (10)$$

where $F(\mathbf{X}^*)$ denotes the objective function value of the global optimum of the search space; w_{4f_3} denotes the weight (penalty) of objective function values between sample minimum Q_1 and lower quartile Q_2; w_{3f_3} denotes the weight of objective function values between lower quartile Q_2 and median Q_3; w_{2f_3} denotes the weight of objective function values between median Q_3 and upper quartile Q_4; w_{1f_3} denotes the weight of objective function values between upper quartile Q_4 and largest observation - sample maximum Q_5; $F(X_{\text{Worst}})$ denotes objective function value of the worst solution (element) of the search space. The evaluation of optimization experiments using the third criterion is shown in Figure 19 and in Figure 20.

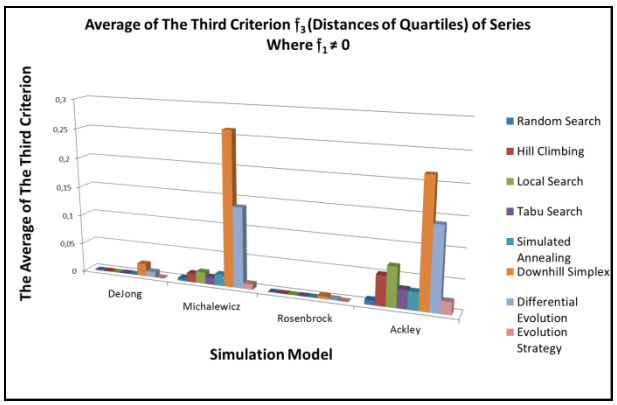

Figure 19: Average of the Third Criterion f_3 - Distances of Quartiles - Testing Functions

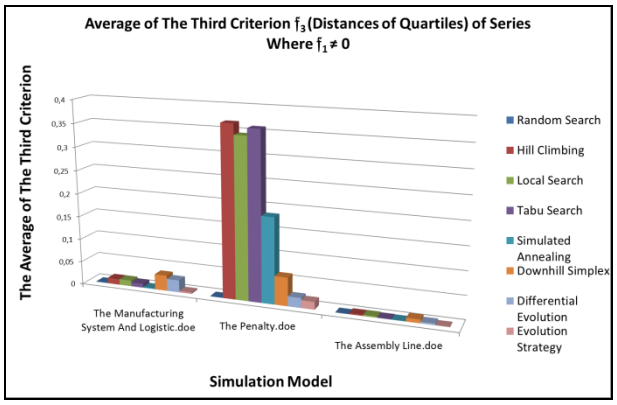

Figure 20: Average of the Third Criterion f_3 - Distances of Quartiles - Discrete Event Simulation Models

The effort is to minimize f_3 ($f_3 \in [0,1]$). If the first criterion equals zero $f_2 = 1$ then the third criterion equals zero $f_3 = 0$ (absolutely successful series). The Downhill Simplex optimization method provided the worst optimization results of all tested optimization methods due to rounding the coordinates. Pseudo gradient optimization methods found solutions of similar quality. Simulated Annealing provides a worse solution than the Evolution Strategy.

6.4. The Number of Simulation Experiments Until the Optimum Was Found

The fourth criterion f_4 evaluates the speed of finding the optimum – the number of performed simulation experiments until the optimum/best solution was found in each series. The effort is to minimize f_4 ($f_4 \in [0,1]$). The fourth criterion when the objective function is minimized can be formulated as follows:

$$f_4 = \frac{|Q_1 - 1| + w_{4f_4}|Q_1 - Q_2| + w_{3f_4}|Q_2 - Q_3| + w_{2f_4}|Q_3 - Q_4| + w_{1f_4}|Q_4 - Q_5|}{m_{\tilde{X}}} \quad (11)$$

where w_{4f_4} denotes the weight (penalty) of number of simulation experiments until the optimum was found between sample minimum Q_1 and lower quartile Q_2; w_{3f_4} denotes the weight of number of simulation experiments until the optimum was found between lower quartile Q_2 and median Q_3; w_{2f_4} denotes the weight of number of simulation experiments until the optimum was found between median Q_3 and upper quartile Q_4; w_{1f_4} denotes the weight of number of simulation experiments until the optimum was found between upper quartile Q_4 and largest observation - sample maximum Q_5; $m_{\tilde{X}}$ denotes the number of feasible solutions in the search space. The evaluation of optimization experiments using the third criterion is shown in Figure 21 and in Figure 22.

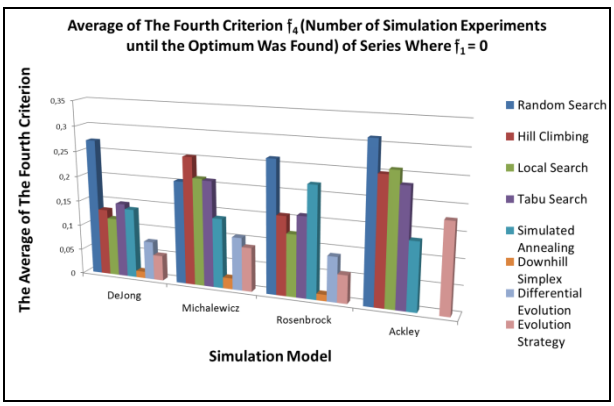

Figure 21: Average of the Fourth Criterion f_4 - Number of Simulation Experiments until the Optimum Was Found - Testing Functions

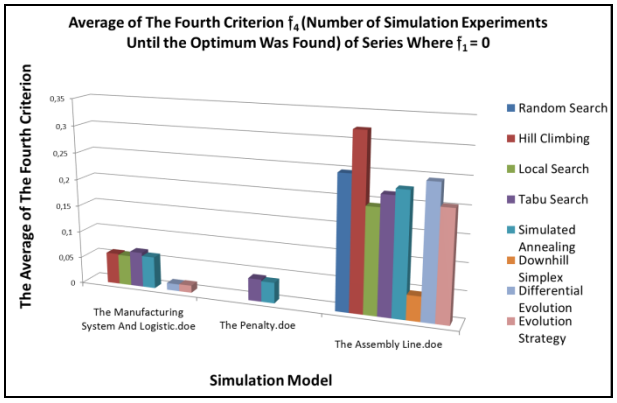

Figure 22: Average of The Fourth Criterion f_4 - Discrete Event Simulation Models

7. CONCLUSION

The goal of our research is to compare selected modified optimization methods (Random Search, Hill Climbing, Tabu Search, Local Search, Downhill Simplex, Simulated Annealing, Differential Evolution and Evolution Strategy) used in the developed simulation optimizer and used in the second module which is focused on testing the implemented optimization methods. Optimization methods generate whole populations instead of one possible solution which prevents premature convergence. The success of optimization methods depends on the objective function landscape. Evolution Strategy is a suitable optimization method for all the tested objective functions (a little propensity to bad tuning of the method parameters). This optimization method achieves good values for specified criteria. The alternative to Evolution Strategy optimization methods is Simulated Annealing. Simulated Annealing has the ability to escape from the local extreme thanks to the implemented approach of setting the temperature to the initial temperature. We can expect to find good results using Random Search if there is a small search space. If the dimension of the search space is bigger, there is little probability of success. Optimization methods based on pseudo-gradient searching such as Hill-Climbing, Local Search, Tabu Search achieve almost the same results for the simple objective function landscape due to their similar nature. Differential Evolution avoids repressing the diversity of solutions (elitism - an advantage of this approach is the faster finding of a feasible solution but not the finding of the global optimum). The range of provided simulation optimization results using this optimization method is better than the optimization methods based on pseudo-gradient searching.

ACKNOWLEDGEMENTS

This paper was created with the subsidy of the project SGS-2012-063 “Integrated production system design as a meta product with use of a multidisciplinary approach and virtual reality” carried out with the support of the Internal Grant Agency of University of West Bohemia and with the subsidy of the Motivation System (POSTDOC) of University of West Bohemia. The paper uses the results of the project CZ.1.07/2.3.00/09.0163 carried out with the support of Ministry of Education, Youth and Sports.

REFERENCES

Koblasa, F., Manlig, F., Vavruska, J., 2013. Evolution Algorithm for Job Shop Scheduling Problem Constrained bythe Optimization Timespan. *Applied Mechanics and Materials*, 309, 350-357.

Kopecek, P, 2012. Heuristic Approach to Job Shop Scheduling. *DAAAM International Scientific Book 2012*, pp. 573-584. October 24-27, Zadar (Croatia).

Michalewicz, Z., Fogel, D. B., 2004. *How to Solve It: Modern Heuristics*, Berlin: Springer,

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.

Monticelli, A.J., Romero, R., Asada, E., 2008. Fundamentals of Tabu Search. In: El-Hawary, M.E., ed. *Modern heuristic optimization techniques*. New Jersey: John Wiley & Sons, 101–120.

Monticelli, A.J., Romero, R., Asada, E., 2008. Fundamentals of Simulated Annealing. In: El-Hawary, M.E., ed. *Modern heuristic optimization techniques*. New Jersey: John Wiley & Sons, 123–144.

Pohlheim, H., 2006. *Genetic and Evolutionary Algorithm Toolbox for use with MATLAB*. GEATbx. Available from: http://www.geatbx.com/docu/fcnindex-01.html [accessed 20 November 2011]

Tvrdik, J., 2004. *Evolutionary Algorithms - textbook*. Virtual information center for Ph.D. students, Ostrava University. Available from: http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf [accessed 6 February, 2011]

Votava, V., Ulrych, Z., Edl, M., Korecky, M., Trkovsky, V., 2008. Analysis and Optimization of Complex Small-lot Production in new Manufacturing Facilities Based on Discrete Simulation. *Proceedings of 20th European Modeling & Simulation Symposium EMSS 2008*, pp. 198-203. September 17-19, Campora San Giovanni (Amantea, Italy).

Weise, T., 2009. *Global Optimization Algorithms - Theory and Application 2nd Edition*. Thomas Weise - Projects. Available from: http://www.it-weise.de/projects/book.pdf [accessed 2 February, 2012]

Wong, K.P., Dong, Z.Y., 2008. Differential Evolution. In: El-Hawary, M.E., ed. *Modern heuristic optimization techniques*. New Jersey: John Wiley & Sons, 171–186.