Proc. IEEE Int. Systems Conference (SysCon 2015), Vancouver, Canada, April 2015, pp. 139-144.

Extending design space optimization heuristics for use with Stochastic Colored Petri nets

Christoph Bodenstein
Systems & Software Engineering
Ilmenau University of Technology
P.O. Box 100 565
98684 Ilmenau, Germany
Email: Christoph.Bodenstein@tu-ilmenau.de

Armin Zimmermann
Systems & Software Engineering
Ilmenau University of Technology
P.O. Box 100 565
98684 Ilmenau, Germany
Email: Armin.Zimmermann@tu-ilmenau.de

***Abstract*—Automatic design optimization of complex systems is a time-consuming task. This paper presents multiphase variants of well-known heuristics for an efficient indirect system optimization with simulation. The tradeoff between accuracy and achievable speedup is analyzed for examples including a benchmark function as well as simulated values for a stochastic colored Petri net application. The analysis are carried out with TOE (TimeNET optimization environment), a software prototype implemented for this task recently.**

I. Introduction

Designing complex embedded systems is a challenging task. Modeling those systems in simulation tools helps to evaluate solutions and to find better ones. Discovering an optimal system configuration is one of the main targets in system design.

Stochastic Colored Petri nets (SCPN, [18], [19]) are a modeling class that is especially useful for the compact description of complex system behavior, while still being based on a formal specification. To find optimal solutions based on such stochastic models, the common approach is a black-box (or indirect) optimization [3], [6], [8], [9], [12] as depicted in Figure 1.

The model typically has several numerical parameters which can be configured. The number of possible configurations defined by each parameter's range and discretization step determines the overall size of the design space, which can be huge. For an exhaustive search for the optimal parameter set, a simulation of the model would have to be started for each of these configurations, resulting in an unacceptable overall run time.

A black-box optimization heuristic starts with a set of parameters (system configuration), evaluates the result of a simulation run, and calculates the next parameter set based on it afterward. The number of simulations is reduced to speed up the search for an optimum, which is paid for by an increasing uncertainty of how well the actual optimum is hit.

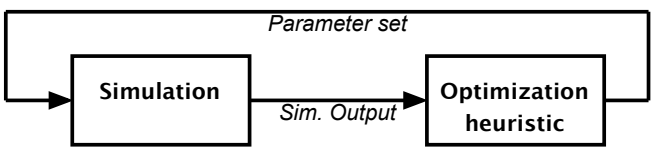

Fig. 1. Common black box optimization, see [6]

The main target of optimization heuristics is to find the best solution with a high probability as fast as possible. There is obviously a trade-off between these two conflicting goals. The time needed for an optimization depends mostly on two factors: the CPU time per simulation run, and the number of necessary simulations. While the CPU time per simulation mainly depends on simulation precision and type of performance measure, in this paper we focus on reducing the number of simulations and improving the quality of the found optima by using a multiphase optimization approach. This is one step in our overall goal to integrate simulation and optimization heuristic more closely to control the tradeoff between speed and accuracy better. Another direction reduces the model's level of detail and/or simulation precision during an optimization as proposed earlier in [21].

In the literature, indirect optimization schemes with two phases are considered mainly to find a promising region of the surface first, which is refined in a second run. This can be done in several ways, including a parameter step size increase (discretization accuracy) or using a faster model approximation in the first phase [14], [16], [22], among others.

If this idea has the general advantage of achieving a comparable optimization accuracy with fewer simulation runs as reported in the literature, considering more than two phases may have the potential of achieving even better performance. This approach has not yet received much attention and is thus followed in this paper. In [1], a particle swarm optimization approach is modified such that groups of particles follow different goals in several phases of the algorithm. In a different type of approach, multiple phases are used to find a set of optima (multi-modal optimization) [2], [13], [17].

Using multiple phases means starting with a coarse discretization of definition space and refining this with every phase until a defined minimum is reached. This is somehow similar to simulated annealing but without the randomness in calculating following parameter sets depending on the chosen base heuristic.

The combination of known heuristics to improve optimization results is part of different research projects and sometimes termed hyper-heuristics [5]. The introduced approach can be

seen as a kind of combined heuristic while only its configuration is changed between phases instead of the complete algorithm.

Automated optimization would not be feasible without an appropriate software tool implementation. The TOE tool used in this paper is currently under development by the first author and described in [4]. It is used in combination with TimeNET, a tool for modeling and simulation of stochastic Petri nets [20]. TOE's architecture is similar to the tool presented in [17]. However, it was not possible to reuse it because of our need for multiple phases of optimization, variation of simulation precision, and interfacing with TimeNET, which in the end made it necessary to develop an independent application. The TOE tool is planned to be available via the TimeNET download site at `http://www.tu-ilmenau.de/timenet` by the end of 2014.

II. Cost Functions in SCPNs

In SCPNs the cost function to be minimized is defined by a performance measurement [15]. The optimization heuristic aims to minimize the distance of this measurement to a given target value or simply to minimize (or maximize) the measurement value itself. Measurements in SCPNs can be probabilities, average token counts, average transition throughput, or combinations of them. These measurements are used as cost functions for optimization, and their shape (or response surface) has a significant impact on the efficiency of optimization heuristics. For the later testing of the heuristics, we developed a model with a non-trivial surface shown in Figure 2. It has a diagonal area of near-optimal (minimal) solutions, with the actual global optimum in one of the corners.

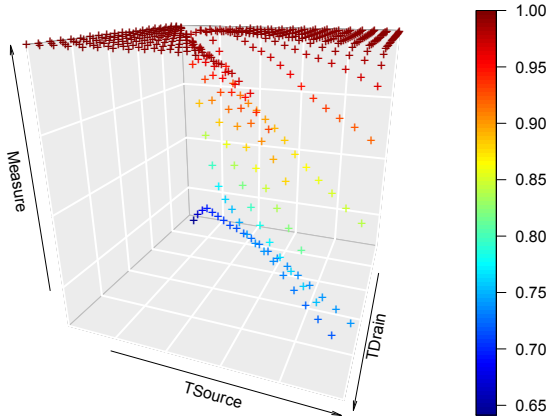

Fig. 2. Response Surface Plot for an Example SCPN

Numerous simulation runs would be necessary to test the efficiency of optimization heuristics and to compare their result accuracy against the actual optimum, because the latter will require a full coverage of the design space. However, to test the heuristics themselves, only the results of the model evaluations are needed. We consider two main techniques for speeding up optimization test runs, which have been implemented in the TOE tool [4].

The first possibility is an "'offline"' optimization. The simulation tool iterates through all possible parameter configurations once and stores the results in a file. The plot of a resulting surface in Figure 2 is an example result. Such a result file can be loaded for subsequent optimization runs and is used to return simulation results by looking them up in the "'result cache"' instead of starting a new simulation each time time. As real simulation runs may need minutes or even hours to finish, this leads to a significant speedup. However, only pre-simulated parameter configurations can be returned to the optimization heuristic algorithm. If a parameter set is requested which is not already stored in the cache (which is only possible if the discretization step is chosen smaller than for the full scan), the next closest ones can be used with an interpolation, with the disadvantage of additional uncertainty.

Apart from that it allows the optimization software to look up the global optimum to evaluate the achieved quality of any optimum found in a run. Both the distance to the real global optimum in the definition range (parameters / design space), and value range (result quality) are characteristic results.

The second method for a fast comparison of heuristics is to use numerically computable benchmark functions instead of a real simulation. Every requested cost function value can be calculated rapidly if necessary, after a mapping of the benchmark to the definition range of the parameters. A comprehensive overview of useful benchmark functions is given in [11], while [7] applies several benchmarks to evolutionary optimization methods.

The so-called *Matyas function* has been chosen from this list as a benchmark for this paper. A detailed plot of its shape is shown in Figure 3.

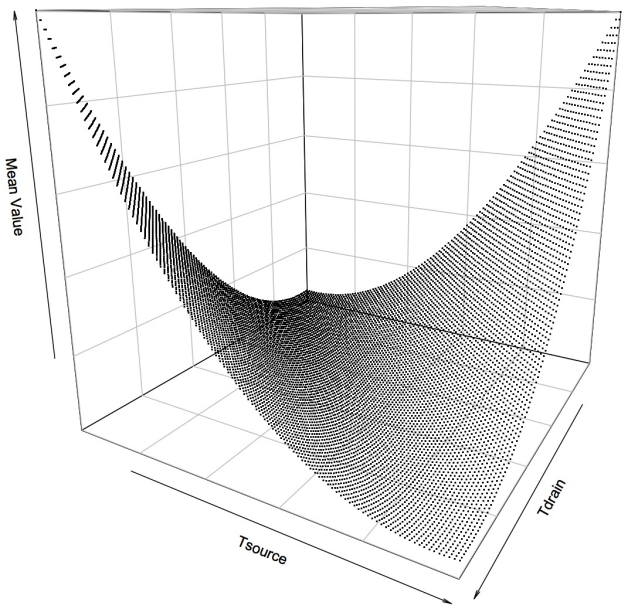

Fig. 3. Standard design space resolution (Matya)

In the following, we aim at evaluating how well certain heuristics perform. As this will probably depend on the type

of response surface, and both mentioned techniques to speed up optimization runs should be used, a coarse-grained full evaluation of the design space has been carried out to estimate the surface shape. An appropriate benchmark function with a similar shape is then configured and used to determine a possible optimization heuristic and its configuration. This heuristic is used in conjunction with a detailed simulation of the actual SCPN model finally.

III. Multi-Phase Heuristic Optimization

If the design space of a modeled system is large and the expected cost function shape is smooth with few local optima, it is a well-known technique to use a pre-optimization to find areas of interest in the design space. Similar approaches have been published in the literature as already described in the introduction. A two-phase optimization strategy is typically used. A coarse optimization should find an interesting area of design space during the first phase, while another (more detailed) heuristic is used to find the optimal parameter set in the second phase. The second phase is typically deterministic while the first phase aims at finding the interesting area with stochastic optimization methods. Another idea is to use simulated annealing in both phases but with different configurations [22].

In our approach we want to find out whether fewer overall simulation runs are necessary if the number of phases is increased even more. Instead of applying different heuristics in every phase or different configurations the first approach is to apply the same heuristic on a reduced design space. This is achieved by different resolutions (step sizes) of parameters in the phases, thus reducing the possible size of design space in the beginning and increasing accuracy in later phases.

Let's assume that a parameter p_i may have values in the interval $p_i \in [0, 999]$, and has an original resolution of 1. In a four-phase optimization attempt, we reduce this resolution to $2^{4-1} = 8$ in the first phase, while the limit of 0..999 is still used. After the first phase the parameter value limits are reduced by half to 500 while the algorithm starts with the found optimum parameter set from phase one. The resolution is doubled for the next phase. As an example, Figure 4 shows the starting point for a four-phase optimization. The coarse resolution allows a more efficient optimization with fewer simulation runs in contrast to a single-phase optimization based on the fine resolution shown in Figure 3.

In the last phase, the resolution is the same as originally specified (e.g. 1 for our example), while the parameter value limits are $1000/2^{4-1} = 62.5$ around the found optimum from phase 3.

The key characteristics of this simplified multiphase approach are

- higher number of phases
- same algorithm in every phase
- reduced design space size (parameter resolution) depending on phase count

This is of course only starting point out of many possible ways to implement the general idea of multiphase optimization, and in the future we will implement and evaluate and compare further approaches that change other aspects of the heuristics per phase.

With several experiments we tried to find out if this approach of reducing the design space resolution while using the same configured heuristics throughout all phases results in a reduced number of necessary simulation runs to find the optimal solution. Moreover we are interested in the probability of finding the optimal solution and the average distance to the calculated optimum to evaluate the quality and achievable speedup of optimization heuristics.

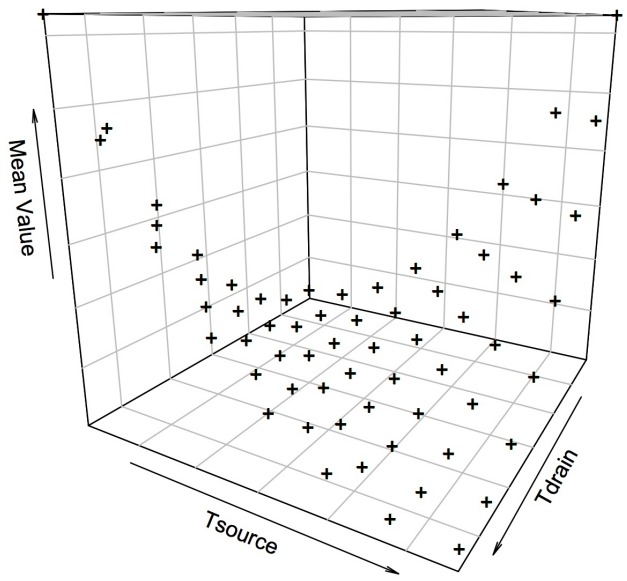

Fig. 4. Design space resolution in phase 1

IV. Experimental setup

The already mentioned software TimeNET Optimization Environment (TOE) was used and modified for the experiments of this paper. Numerical benchmark functions were implemented in addition to the existing standard simulation, to calculate cost functions for rapid testing. These functions get the normal parameter set from the optimization heuristic as input just as if a real simulation would have been started. Parameter values are mapped into the definition range of the benchmark function, which for our example results in a cost function shape as shown in Figure 3.

A resolution of 10,000 possible values was chosen for each of the two parameters that the Matyas function is defined for. The global optimum for this function is known to be at 0.0, the exact middle of the possible parameter ranges. Therefore the distance of found optima in definition range and value range can be calculated. This can be important because some cost functions have optima with very similar values but large distance in definition space.

The tool has been extended to support repeated runs of whole optimization experiments, to calculate the probability of finding an optimum and the average count of necessary simulation runs. Every optimization heuristic was run 100 times to average over the inherent nondeterminism in the heuristic and simulations, using differing random number seeds.

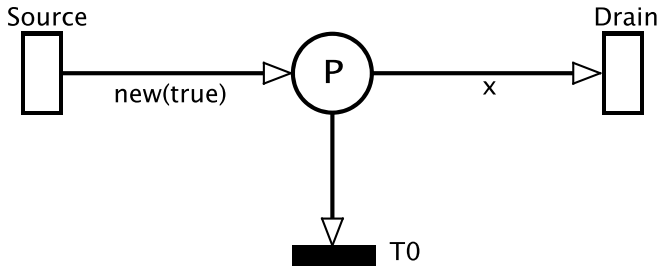

Fig. 5. Used SCPN for optimization tests

Hill climbing and *simulated annealing* have been chosen as base heuristics for evaluating the multiphase approach. Both are well known, and numerous different implementations exist. The same configuration was used in single-phase and multiphase optimization to achieve a fair comparison between the approaches.

The hill-climbing algorithm calculates the next parameter set by increasing the value of one parameter. If the cost function result is worse than the last one, it continues by trying to change the next parameter. It terminates if the result could not be improved within four trials. These values lead to an acceptable tradeoff between number of simulations and probability of finding the optimum for our example, and have been fixed to make the experiments comparable.

Simulated annealing calculates the next parameter set by varying all changeable parameters at once, while the distance of a parameter to the previous value is defined randomly and scaled by a temperature parameter. Temporarily worse solutions are accepted with a certain probability that also depends on the temperature. The temperature decreases over time, thus emulating physical processes such as crystallization. Fast Annealing [10] was chosen as cooling method for both parameters, and the algorithm variant described in [19] was implemented. This (and other implementations) work with a continuous definition range of floating-point numbers for parameters. We use a discretized definition range that simply rounds to the next possible parameter set.

In the real SCPN model, two parameters can be iterated in the range $[1, 1001]$. With a resolution of 10, all possible variations are simulated to be used in "offline" optimization runs, and stored in a file for later caching first. The used SCPN is shown in figure 5. Source and Drain are the main transitions. The parameters *Tsource* and *Tdrain* determine the firing rates in the form of $F = EXP(\frac{1.0}{Tsource})$ and $F = EXP(\frac{1.0}{Tdrain})$. T0 fires immediately if the number of tokens in P is greater then 70.

$$Measure = 1 - ((40 \leq \#P)\&(50 \geq \#P)) \tag{1}$$

The cost function is defined as in eq. 1. So the measure describes the empirical probability that P does not contain 40-50 tokens. It is constructed academic example net but serves well as base for optimization tests.

V. Results

To understand the review of optimization results shown in Table I and Figures 6, 7, and 8, some remarks are necessary. For our example, the global optimum of the benchmark functions is known as well as the definition range and value range which can be chosen by the user or developer of an optimization software. In these experiments we chose the Matyas function with a range of $x, y \in [-10, 10]$. Its optimum is exactly at $f(0, 0) = 0$. The maximum value is $f(-10, 10) = 100$. The relative distances of the parameter sets that are found heuristically can thus be calculated. The values px, py of parameters to be "optimized" are mapped into the definition range of benchmark functions f, shown in eq. 2. Therefore it is not possible to get bigger function values then 100.

$$\begin{aligned} P(px, py) &\rightarrow f(fx, fy) \\ px_{min}, px_{max} &\rightarrow fx_{min}, fx_{max} \end{aligned} \tag{2}$$

The relative distance in the value range equals the distance to 0 in relation to the overall value range of 100 in this case. Relative distance in definition range is calculated as the sum of all differences of the coordinates of the optimum in relation to the sum of all parameter ranges which is 40 here.

Opposed to the benchmark function, an exact calculation of the achievable maximum cost function value or optimum coordinates is not possible. However, as we simulate the model for all possible discretized parameter configurations in advance with sufficient accuracy, the resulting cost function values are known with enough accuracy. For our example, the maximum value is 1.0, while the optimum is 0.63. The resulting range of 0.37 is used as the base to calculate the relative distance of any optimum found by a heuristic in the value range. The definition range is $(1, 1001)$, and used to calculate the relative distance in for parameters.

The number of simulations needed to find an optimum is an important result for our evaluation, but requires a stop condition in the heuristic that is usually based on the actual cost function values. Hill climbing is an example for such an algorithm. As described earlier, it accepts the measured value as the optimum if the four next possible parameter sets did not improve the result.

Simulated annealing, on the other hand, may accept a new parameter set even if a worse result is calculated for it. The probability is based on the decreasing temperature and converges to zero over time. In our experiments we used the algorithm for simulated annealing as provided in [19] together with a Fast annealing cooling function. The temperature for calculating the next parameter set as well as the temperature for accepting a worse cost function value is calculated as shown in Formula (3).

$$T_k = \frac{1}{k} * T_0 \tag{3}$$

In this way the number of simulations is increasing with the number of phases to be used because in every phase the same start and end temperature as well as the cooling function and abort condition is used. It can act as a reference value for the increasing number of simulations when hill climbing is used. While both heuristics show a steady increasing number of simulations hill climbing tends to need many more simulation runs if more then 8 phases are used. Currently this

behavior cannot be explained and will be treated in further investigations.

Every optimization heuristic and configuration was tested 100 times, starting with random parameter sets. The repeated execution of optimization allows to calculate the empirical probability of finding the optimum for each heuristic. Moreover, it may be interesting to see if the resulting "'optimal"' parameter sets are within a certain area around the real optimum. This area can be either a radius of n% around the cost function results, or around the parameter values of this calculated optimum.

A possible measure for the accuracy of a specific heuristic is to check how large such a radius has to be to contain 90% of all optima found by the heuristic. The results are shown in Table I. As an example, 14.84 simulation runs were needed on average to find an optimum while the relative distance to the known absolute optimum is 2.4% for hill-climbing. If we would assume a radius of 4% around the absolute optimum (which means all values from 0 to 4.0 in Matyas function), 90% of all found optima would be accepted.

Heuristic	**∅ Sim-#**	**∅ Distance**	**90% Radius**
HillClimbing	14.84	2.4%	4%
2-phase, HillClimbing	23.92	6.6%	21%
4-phase, HillClimbing	31.55	5.4%	10%
6-phase, HillClimbing	47.56	5.4%	14%
8-phase, HillClimbing	587.6	8.0%	34%
10-phase, HillClimbing	581.8	5.0%	15%
Simulated Annealing	12	2.7%	4%
2-phase, Sim. Annealing	25	4.35%	6%
4-phase, Sim. Annealing	49	4.6%	11%
6-phase, Sim. Annealing	73	6.0%	14%
8-phase, Sim. Annealing	97	5.0%	11%
10-phase, Sim. Annealing	121	5.25%	11%

TABLE I. RESULTS OF MULTIPHASE OPTIMIZATION EXPERIMENT WITH BENCHMARK FUNCTION

Figure 7 shows the average relative distance of found optima to the real optimum in value range (i.e., the quality of the optimized value). When using the benchmark function as cost function, increasing the number of phases does not affect the results. Yet, it is interesting to see that, for real SCPN simulations, both heuristics perform significantly better when increasing the number of phases until 4, compared to the optimization with one phase. More than 4 phases, however, seem to be counterproductive in this context.

Another evaluation looks at the distance to the known optimum in the definition range. In many cases this is unnecessary because the optimal (mostly minimal) value of cost function is more important than the configured parameter values. However, for many technical systems it is interesting to see if parameter configurations with similar resulting function values are nearby or completely different. This may be a sign of robustness or chaotic system dependence on parameters (sensitivity). Figure 2 shows an example where two possible configurations result in very small cost function values close to the optimum, but with completely different parameter configurations, each on the opposite side of definition space. Another example for this behavior is the Eggholder benchmark function which has many local optima but only one global [11].

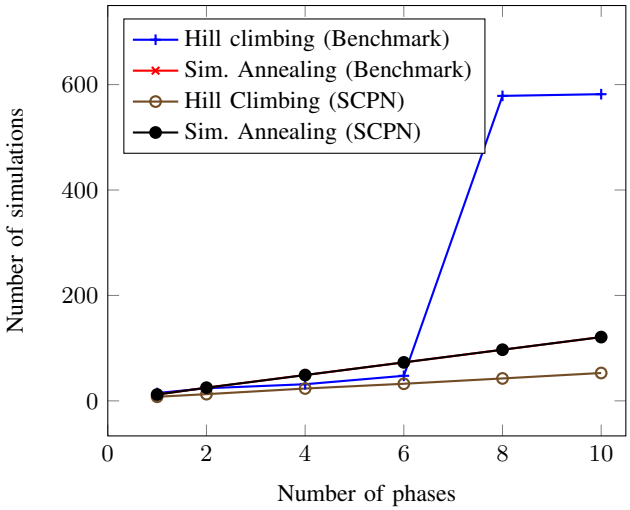

Fig. 6. Number of optimization phases vs. Number of simulation runs

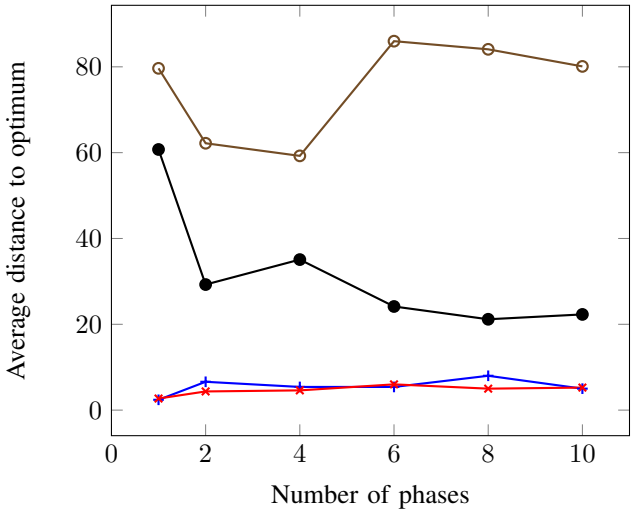

Fig. 7. Number of optimization phases vs. Average distance to optimum in value range

The results in Figure 8 show an improvement of optimization results regarding the distance in definition range when using Matyas function as benchmark. Increasing the number of phases to 4 enhances the results dramatically. Hill climbing performs considerably better than simulated annealing.

Using the pre-simulated data from SCPN simulations leads to quite different results: Increasing the number of phases only leads to worse results. The reasons for this unexpected behavior are currently being further investigated. Maybe the coarse discretization in the definition range caused misleading results, such that the experiments will have to be repeated with finer discretization or even continuous definition range and "online" simulations.

VI. CONCLUSION

In this paper an approach to improve simulation-based optimization by applying well-known heuristics in multiple phases is described. The effect on optimization result quality of

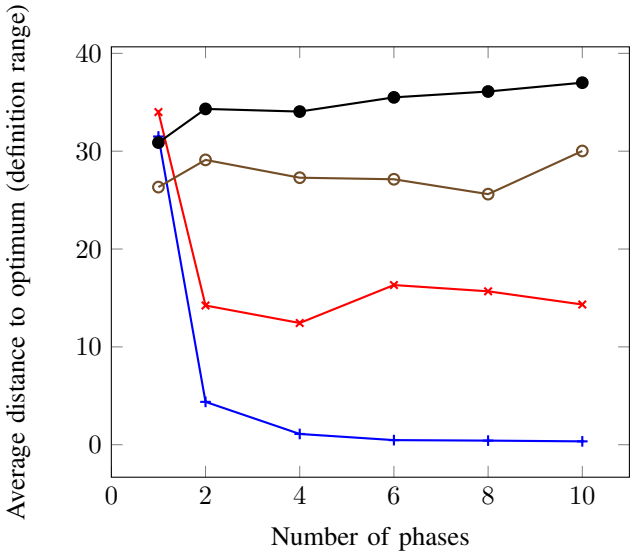

Fig. 8. Number of optimization phases vs. Average distance to optimum in definition range

simulated annealing and hill climbing in a multiphase meta-heuristic are compared. The quality of the found optima is evaluated via their distance to the known global optimum, both in value range and definition range. All algorithms were tested on the Matyas benchmark function as well as for pre-calculated SCPN simulations.

Evaluating the distance to absolute optimum in definition space showed that increasing the number of optimization phases leads to much better optimization results especially for hill climbing. However, using more then 4 phases did not improve the optimization results for both used heuristics. On the other hand, the experiments show that increasing the number of optimization phases will not reduce the number of necessary simulation runs directly, and also will not improve the quality of found optima in terms of distance in value range, especially for the used benchmark function.

So far, no improvement could be detected when increasing the number of phases for SCPN pre-simulated results. As the coarse discretization of the used measurement data could be a cause for that, further investigations are currently undertaken to understand the reasons for this behavior. Moreover, other heuristics for use in multiple phase optimization and their influence on result accuracy or speed will be tested in our future work.

ACKNOWLEDGMENT

This paper is based on work funded by the Federal Ministry for Education and Research of Germany with grant number 01S13031A.

REFERENCES

[1] B. Al-kazemi and C. Mohan. Multi-phase generalization of the particle swarm optimization algorithm. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 489–494, May 2002.

[2] A. Beghi, M. Bertinato, L. Cecchinato, and M. Rampazzo. A multi-phase genetic algorithm for the efficient management of multi-chiller systems. In *Asian Control Conference, 2009. ASCC 2009. 7th*, pages 1685–1690, Aug 2009.

[3] J. Biel, E. Macias, and M. Perez de la Parte. Simulation-based optimization for the design of discrete event systems modeled by parametric Petri nets. In *Computer Modeling and Simulation (EMS), 2011 Fifth UKSim European Symposium on*, pages 150–155, 2011.

[4] C. Bodenstein and A. Zimmermann. TimeNET optimization environment. In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, 2014.

[5] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics. *J Oper Res Soc*, 64(12):1695–1724, Dec 2013.

[6] Y. Carson and A. Maria. Simulation optimization: Methods and applications. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 118–126, 1997.

[7] J. M. Dieterich and B. Hartke. Empirical review of standard benchmark functions using evolutionary global optimization. *CoRR*, abs/1207.4318, 2012.

[8] M. C. Fu. Optimization via simulation: a review. *Annals of Operations Research*, pages 199–248, 1994.

[9] M. C. Fu. A tutorial overview of optimization via discrete-event simulation. In G. Cohen and J.-P. Quadrat, editors, *11th Int. Conf. on Analysis and Optimization of Systems*, volume 199 of *Lecture Notes in Control and Information Sciences*, pages 409–418, Sophia-Antipolis, 1994. Springer-Verlag.

[10] L. Ingber. Adaptive simulated annealing (asa): Lessons learned. *Control and Cybernetics*, 25:33–54, 1996.

[11] M. Jamil and X.-S. Yang. A Literature Survey of Benchmark Functions For Global Optimization Problems. *ArXiv e-prints*, Aug. 2013.

[12] S. Künzli. *Efficient Design Space Exploration for Embedded Systems*. Phd thesis, ETH Zurich, Apr. 2006.

[13] G. Lin, J. Zhang, Y. Liang, and L. Kang. Multi-phase evolutionary algorithm for non-linear programming problems with multiple solutions. In *Neural Networks and Signal Processing, 2008 International Conference on*, pages 382–387, June 2008.

[14] B. L. Nelson, J. Swann, D. Goldsman, and W. Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, 49:950–963, 1999.

[15] W. H. Sanders and J. F. Meyer. A unified approach for specifying measures of performance, dependability, and performability. In A. Avizienis and J. Laprie, editors, *Dependable Computing for Critical Applications*, volume 4 of *Dependable Computing and Fault-Tolerant Systems*, pages 215–237. Springer Verlag, 1991.

[16] F. Schoen. Two-phase methods for global optimization. In *Handbook of global optimization*, pages 151–177. Springer US, 2002.

[17] M. Syrjakow, E. Syrjakow, and H. Szczerbicka. Tool support for performance modeling and optimization. *International Journal of Enterprise Information Systems*, 2005.

[18] A. Zenie. Colored stochastic Petri nets. In *Proc. 1st Int. Workshop on Petri Nets and Performance Models*, pages 262–271, 1985.

[19] A. Zimmermann. *Stochastic Discrete Event Systems - Modeling, Evaluation, Applications*. Springer-Verlag New York Incorporated, Nov. 2007.

[20] A. Zimmermann. Modeling and evaluation of stochastic Petri nets with TimeNET 4.1. In *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th Int. Conf. on*, pages 54–63, Oct 2012.

[21] A. Zimmermann and C. Bodenstein. Towards accuracy-adaptive simulation for efficient design-space optimization. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 1230 –1237, Oct. 2011.

[22] A. Zimmermann, D. Rodriguez, and M. Silva. A two-phase optimisation method for Petri net models of manufacturing systems. *Journal of Intelligent Manufacturing*, 12(5/6):409–420, Oct. 2001. Special issue "Global Optimization Meta-Heuristics for Industrial Systems Design and Management".