Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

FAST CONVERGING, AUTOMATED EXPERIMENT RUNS FOR MATERIAL FLOW SIMULATIONS USING DISTRIBUTED COMPUTING AND COMBINED METAHEURISTICS

Dr. Christoph Laroque

Chair for Modeling and Simulation
Technical University of Dresden

Noethnitzer Straße 46
Dresden, D-01167, Germany

Alexander Klaas
Jan-Hendrik Fischer
Mathis Kuntze
Business Computing, esp. CIM
Heinz Nixdorf Institute,
University of Paderborn
Fuerstenallee 11
Paderborn, D-33102, Germany

ABSTRACT

The analysis of production systems using discrete, event-based simulation is wide spread and generally accepted as a decision support technology. It aims either at the comparison of competitive system designs or the identification of a "best possible" parameter configuration of a simulation model. Here, combinatorial techniques of simulation and optimization methods support the user in finding optimal solutions, but typically result in long computation times, which often prohibits a practical application in industry. This paper presents a fast converging procedure as a combination of heuristic approaches, namely Particle Swarm Optimization and Genetic Algorithm, within a material flow simulation to close this gap. Our integrated implementation allows automated, distributed simulation runs for practical, complex production systems. First results show the proof of concept with a reference model and demonstrate the benefits of combinatorial and parallel processing.

1 MOTIVATION

Modern business computing, especially in the area of operations research, offers a wide variety of methods for complex problem solving for planning, scheduling and control of production and logistic processes. Those processes, which are to be either designed or improved, are typically projected to models and then optimized by the use of simulation and/or optimization technologies in order to improve decision variables and resulting key performance indicators under a given set of restrictions.

In simulation, this improvement is usually achieved by the iterative evaluation of multiple scenarios and their subsequent simulation results (Law and Kelton 2000). In the case of optimization, the optimal configuration is achieved by mathematical optimization algorithms or (meta-) heuristic approaches (Rardin and Ronald 1997).

Due to the high computational demand of both iterative evaluation and mathematical optimization, specific procedures as a combination of both simulation and optimization were derived. They combine both advantages: an optimization algorithm can be used to automatically generate a specific model configuration, which can be evaluated by simulation runs (Fu 2002). Especially for simulation models with stochastic influence factors, which need a high amount of simulation runs, these procedures can lead to faster identification of improving model configurations than standard methods for the design of simulation experiments (Fu 2002).

It remains a challenge to further improve the performance of finding a good solution with a high global quality, especially in the area of production. The given complexity of the underlying systems, and thereby the simulation model, is very high, so that the application of standard combinatorial approaches

of mathematical optimization and material flow simulation is infeasible for industrial applications due to high computation costs.

Our approach uses a combination of heuristics, Particle Swarm Optimization and a Genetic Algorithm, that are employed in a distributed matter. We speed up computation through parallel processing and achieve fast convergence. Our implementation is integrated with the material flow simulator d³fact and manages initiation of nodes and data exchange between them. By using generic interfaces to d³fact, we are able to apply our method in a practical, industrial environment.

The paper presents in short the necessary state of the art in simulation based optimization and design of experiments in the following section. The conceptual approach of the procedure is presented in section 3, followed by the prototypical implementation in the material flow simulation tool d³fact in section 4. The first evaluation results of the procedure are shown in section 5. The paper closes with an outlook on future work in this area.

2 STATE OF THE ART

The following section presents previous approaches regarding simulation based optimization as a design of experiment and using metaheuristics.

2.1 Experimental Design

Design of experiments (DOE) refers to the use of statistical techniques to create an efficient, systematic set of controlled experiments for collecting data in order to estimate relationships between independent and dependent variables through measurement. In the area of simulation, DOE is used for the systematic evaluation of simulation models in order to identify a set of model parameters, which leads to the desired simulation results. Each simulation run hereby evaluates a concrete set of parameters. Typically, the simulation models include stochastic influence factors, so that a single simulation run is not sufficient for the evaluation of the parameter set, and multiple simulation runs for each of the configuration sets are to be performed. Efficient procedures like 2^k-factorial-Design (Banks et al. 2000), Plackett-Burman-experiments as well as response-surface method (RSM) or evolutionary optimization (EVOP) (Fu 2002) are used to reduce the number of required simulation runs by determining parameter sets that will likely lead to a good result.

2.2 Simulation-based Optimization

Simulation-based optimization is an automated DOE method to find a set of globally optimal simulation parameters regarding a given optimization problem. It can be seen as a two stage method, as depicted in Figure 1. In each step, the optimization package outputs a set of simulation parameters that are to be evaluated by the simulation. Based on previous results, the next set of parameters are computed until a stopping criteria is satisfied.

Simulation-based optimization can be seen as a movement through the search space of simulation parameters, where the simulation maps a point in that space to a point in the space of performance indicators. The optimization therefore needs to ensure that the found solution is not just a local optimum, and it needs to find a good solution (optimality is generally not guaranteed) in a small number of simulation runs, i.e. converge quickly (Karibian and Olafsson 2007). Many different types of optimization strategies exist (see (Hachicha et al. 2010) for a classification approach) and have also partially been implemented in commercially available tools (Fu 2002) (Law and Kelton 2000).

We focus on metaheuristics, specifically Genetic Algorithms (Russell and Norvig 2009) and Particle Swarm Optimization (Kennedy and Eberhart 1995). Both methods work generically without knowledge of the concrete problem, making them universally applicable using standard implementations. Each simulation parameter set is mapped to an individual or particle.

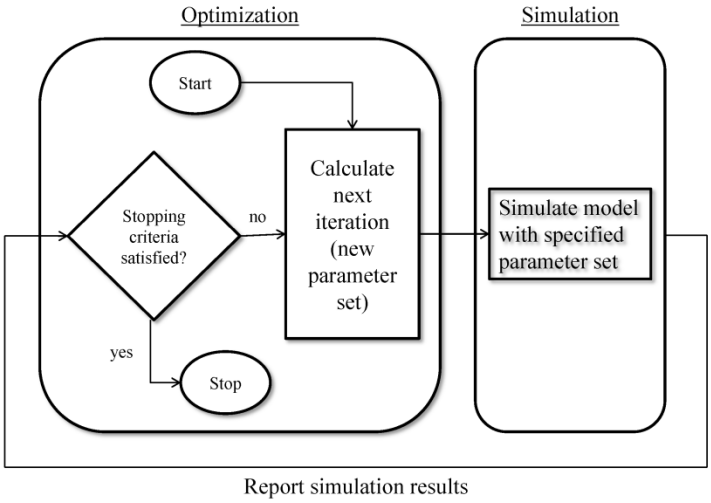

Figure 1: Overview of a simulation-based optimization as an interacting optimization and simulation package. Based on (Law and Kelton 2000).

In the case of Genetic Algorithms (GA), these individuals are selected based on their "fitness" – the quality of a parameter set which the simulation determines, in our case – and form new individuals (a new generation) through crossover and mutation. Similar to natural evolution, fitness is expected to increase over time. Particle swarm optimization improve the set of particles by moving them within the search space based on their position and assigned "velocity", inspired by the way a bird flock or fish school would behave. Advantages of both methods (fast convergence, broad search area) can easily be transferred to the simulation optimization problem.
GAs have been successfully applied to simulation based optimization (Paul and Chanev 1998), (Krug 2002), (Krug et al. 2002) and general implementations of these metaheuristics are available in various open software libraries (e.g. ECJ (Luke 2012)). Some of these solutions offer multiple optimization strategies: the user can select, dependent on a specific use-case, which one to select. An implementation of multiple optimization strategies in parallel in a common distributed environment, where identified solutions are interchanged, are not known.

3 CONCEPT

This work presents an approach to simulation-based optimization using distributed evolutionary computing algorithms with a web-based interface for heuristic's selection and parameter setting. A frequently occurring problem in simulation-based optimization are prohibitively long run times to evaluate a single design point. Further computation time is needed for multiple replications of stochastic simulation models. To be able to optimize complex simulation models, a solution is needed which allows faster evaluation of potential solutions and a fast solution finding process to achieve low overall run times with respect to finding near-optimal solutions. To achieve this, we use distributed islands and (a)synchronous cooperative evolutionary metaheuristics to guide the solution finding process. This concept uses separately working evolutionary metaheuristics, which are able to exchange solutions. Each evolutionary metaheuristic has a master/slave architecture for the parallel evaluation of individuals. A web-based user interface offers the possibility to assign Genetic Algorithms and particle swarm optimization to the islands and to set the respective heuristic's parameters.

3.1 Optimization by Heuristics

In evolutionary metaheuristics, the "fitness" of an individual serves as a good abstraction of the rough position in the solution space of a possible solution to a problem. Fitness values provide rules of thumb as

the main metric for the quality of a solution, as better fitnesses also generally describe "better" situations in the problem domain overall.

With our approach, metaheuristic optimization is done using individual fitness values that are determined through the usage of simulation. Through this, it is possible to map the abstract concept of "fitness" against the functionality of a complex system, e.g. a production system. Metaheuristic individuals generally consist of a limited set of essential numerical data or similarly simple data structures. This information is not sufficient to model the complexity of many systems, given the system behavior in question cannot be itself abstracted easily into closed mathematical form or sufficiently approximated in other ways. This is most likely the case when there are emergent effects inherent in the system that are highly non-linear.

In effect, what our approach is focused on are systems that are sufficiently complex, so that breaking down their behavior into any exact or approximated mathematical form is considerably more difficult than to model it using DES methods and running simulations on it. For modeling, it is generally sufficient to know roughly the input and output characteristics of all contained subsystems or modules. Understanding their interplay is less a requirement but rather a result of doing simulation.

On the other hand, using metaheuristics can yield large combinatorial advantages for the search in the solution space to a given problem. If properly applied, metaheuristics enable a somewhat directed and only semi-random (evolutionary, *natural*) traversal of solution spaces using diversification and intensification (Blum 2003). Relatively "good" solutions can be found faster than when using more exhaustive methods (e.g., MILP, brute force), in addition to being more easily applicable to a range of problems than problem specific heuristics, e.g., a tailored one-use-only local search heuristic algorithm that approximates specific system behavior.

There are also various technical advantages, some of which we emphasis in our approach. The fact that individuals are semi-random data structures, mostly independent from each other, allows us to evaluate a number of them at the same time (*in parallel*). The single notion of *order* is given through the generations. Only once at the end of every generation, the comparison and selection of good individuals and the creation of new individuals has to occur in an orderly, centralized fashion. So, parallel evaluation of many individuals belonging to the *same generation* is apparently easily possible. This promises big advantages regarding computational times. Here, parallel metaheuristic evaluation can be seen as a special case of parallel computing where a deliberate increase of the *problem size* (here: population sizes, generations) will in the end lead to *better* results *faster* than just solving one problem instance at a time (Cung et al. 2002). Many variants are evaluated and compared with each other per time unit, leading to a potential decrease in overall convergence time. In addition, through the use of the island concept, entirely different paths of solution search can be pursued at the same time.

3.2 Webbased User Interface

Setting and testing the heuristic parameters takes often a considerable amount of time during the development of problem and case specific solutions. These parameters have a large influence on the heuristic‘s behavior such as exploration of the solution space or premature convergence in local optima and should be therefore fit for the problem (Talbi 2009). The repetitive tasks of this process are configuration, execution, output analysis and documentation and often not integrated in one tool and must be done in parameter text files or source code. In a distributed island environment, parameters must be set and send for each run to each participating machine. After the execution the results must be collected and analyzed. Due to the fact, that all executions are distributed on multiple machines, it complicates the output analysis of individual machines. A solution is needed that integrates all these aspects and reduces time and effort for these tasks. Our proposed approach uses a web-based interface for a server which controls the complete process. It offers the possibility to set parameters, assign machines and number of threads to use for each machine. After initializing the run, the execution can be observed in an aggregated overview on the outputs of all machines. The parameters, outputs and results for each are stored in a database for later analysis and reuse.

4 IMPLEMENTATION

In our approach, we specifically use the metaheuristic concepts of Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and the island concept. Additionally, we make use of the parallel evaluation of individuals, as discussed in the concept chapter 3.1. We looked at different existing frameworks that allow for optimization modeling using metaheuristics and the features we desired. The one that came closest to our considerations was the ECJ Framework. This is a software framework with an extensive range of functionalities in the field of metaheuristics. ECJ comes with many well-known concepts already pre-implemented, including genetic algorithms and the island concept. It also already contained a semi-functional implementation for PSO which we could easily complete and adapt to our needs.

As mentioned earlier, one of the key aspects of our implementation is to explicitly make use of parallelism when evaluating individuals. ECJ supports this directly, as it provides a robust master-slave architecture for distributed evaluation. We integrated a material flow simulation into ECJ to determine the individual's fitness value. A Microsoft IIS Application Server hosts the control website and manages the virtual machines for the island models.

4.1 Optimization by Heuristics

In contrast to standalone operation, master-slave evaluation consists of a master process that controls the creation of generations through the evolution of prior individuals. In effect, it holds the complete state of an optimization run. It does not directly evaluate newly created individuals, though. They are rather sent to a number of slave processes which may be running on an arbitrary machine. Communication takes place over a standard TCP/IP channel using Java sockets.

To include the d³fact DES kernel in the evaluation, we put it inside of the evaluation function of these slave processes (see Figure 2). Every time a slave starts to evaluate an individual sent to it by the master process, a DES simulation is started using the individual's parameter vector. The slave runs this simulation completely non-threaded and is blocked until the run is completed. It then takes the solution of the run and puts it as the individuals fitness value before it sends it back to the master as the result. It also unblocks and waits for another individual. The waiting time is very short in most cases, provided the current generation at the master is not already completely distributed to slaves. If not, the slave must wait until all other slaves that are still evaluating (i.e., simulating) finish their work. These are the only cases when the overall parallelization is not constantly n-fold, given n slave processes.

Because evaluation does not generally involve simulating a complex model for several seconds like in our approach, ECJ offers techniques to overcome delays and waiting times caused by communication. Several individuals can be bundled and sent as a batch job and waiting times at the slaves could also be used for distributed evolutionary steps using all individuals from a batch job. Those features can most certainly improve average delays and throughputs in many cases. But in our case, they would be unnecessary or counterproductive. Sending only single individuals and keeping all state centralized at the master ensures that waiting times at the end of a generation are minimized. Otherwise, one of the slaves could be backlogged with several individuals and slow down the whole optimization considerably. Because the effort for generating and transmitting individuals is negligible compared with the simulation runs, single individuals can always be sent as "late" as possible. Through this, there is no backlogging at slaves and the evaluation is slowed down less near the end of a generation, which does not end until the last individual has been evaluated.

Our observations show that this approach yields considerable benefits in terms of runtimes. The speedup rate of solving a problem in parallel is given as:

$$Speedup(n) = \frac{runtime\ with\ 1\ CPU}{runtime\ with\ n\ CPUs}$$

What follows is that a (theoretical) perfect parallelism of 100% will lead to an n-th of the single-threaded runtime, given the initial problem size stays the same over every level of parallelization. Test runs for our exemplary model in fact show distinct rates of actual parallelism and speedup.

Observed speedup rates for fixed problem size of 20 individuals per generation:

- for 4 CPUs: 3.2
- for 8 CPUs: 5.2

These figures imply an effective parallelism of over 90%, according to Amdahl's Law (Amdahl 1967). We conclude from this that our approach at parallelization works as expected. On average, processes run simulations in parallel for more than 90% of the time, which allows the whole evaluation of a generation to scale well by adding slave processes on additional processors.

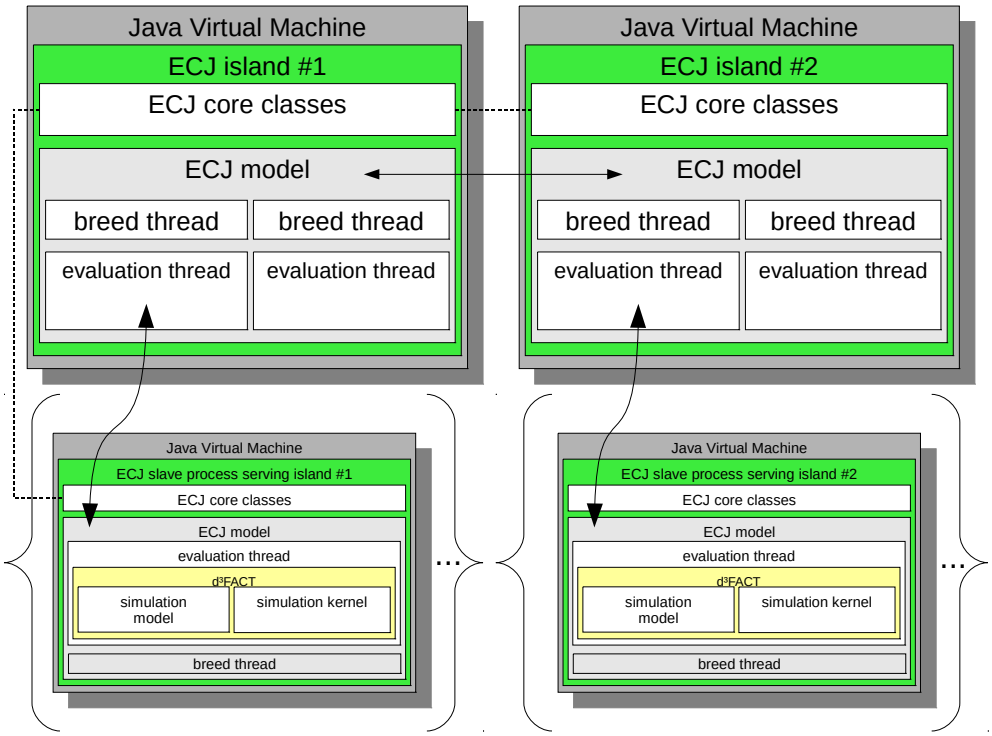

Figure 2: master-slave scheme used within two islands

Parallelizing the evaluation to speed it up is not the only instance of parallelism we worked with, though. We also made use of the island concept which also constitutes a form of parallelism, but rather on a conceptual level of the metaheuristics themselves, as illustrated in Figure 3. Instead of having one population with a set of individuals and one evolutionary “tree”, optimization using islands makes use of several of these. This enables multiple varying evolutions and, in effect, ways of movement through the solution space in parallel. These metaheuristic sub-runs function in isolation from one another to a certain extent. Once every number of generations, the best individual of each island is sent over to other islands where they might be incorporated into the “local” population and have some impact on the convergence of the best found solution there.

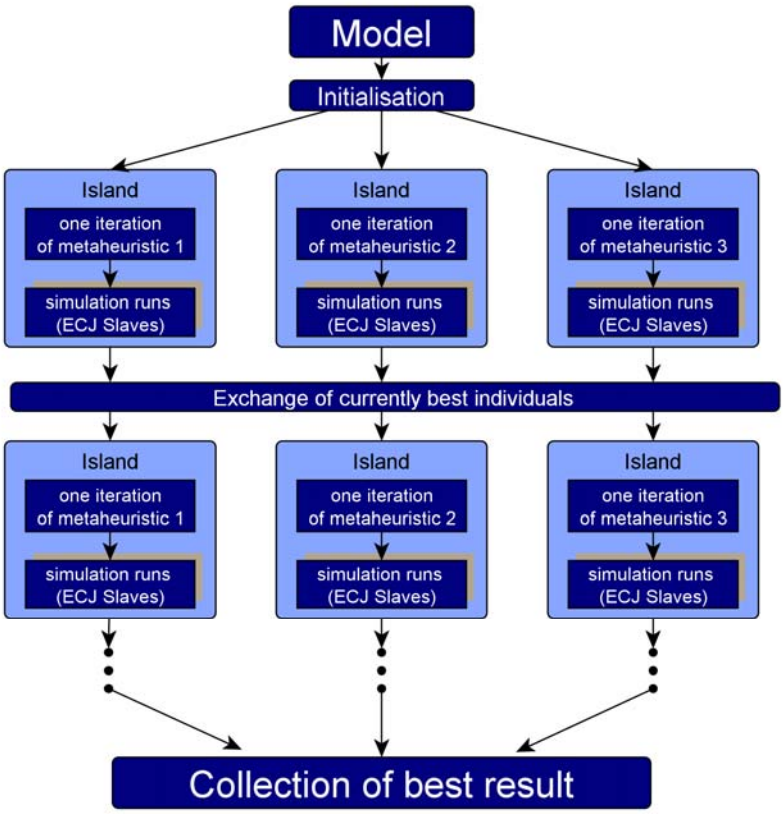

Figure 3: Overview of our parallel, distributed optimization scheme using metaheuristics

Both this, having sub-runs at all and periodically sending individuals, brings together two advantages. One is the diversity through differentiation in evolutionary paths and, through this, a better chance to avoid hitting the same "good" local optima, a common problem of metaheuristics. The other is agility by the introduction of individuals of a wholly different evolutionary process. Metaheuristics like the PSO or Simulated Annealing show a strong convergence behavior after some time of running, which is a deliberate part of their operating principle. Having different instances of this convergence behavior and the sharing of good solutions found by these instances is a promising approach to improve the expectable solution quality of an optimization run.

We combined ECJ's modules for the island concept with its master-slave parallelization to combine both forms of parallelism and their advantages, speed and diversity. Every island is also a master process of its own and has its own pool of slave processes. An island is responsible for breeding and selecting from its own population and distributing every generation among its slaves for evaluation. In the results chapter, we examine the behavior of island processes with different configurations of metaheuristics on the basis of simple ECJ models.

4.2 DES-Simulation with d³fact

d³fact is a discrete, event –based material flow simulation framework, designed and implemented at the Heinz Nixdorf Institute of the University of Paderborn, Germany. Designed as a multi-user environment, it allows simultaneous, collaborative modeling and simulation of a model by multiple simulation experts. d³fact consists of a modeling tool, a simulation server, that runs the simulation and few visualization options from 2D to 3D. The freeware software is based on the Eclipse Rich Client Platform (RCP) and is implemented in Java (Dangelmaier, Huber, Laroque, and Mueck 2005), (Laroque 2007). The project pro-

vides a Java API to program material flow simulation models independently from the modeling interface.

In order to integrate the material flow simulation with ECJ, it suffices to implement the necessary „Problem“ interface and class. The simulation model is built on the individual‘s evaluation step according to the individual‘s position. The position of the individual is a vector which represents a possible design point of the simulation model. The simulation is used to determine the fitness value for this parameter set by generating the model and simulating it. After generating the simulation model automatically based on the design point, the simulation run is performed. In case of stochastic influences, several replications are done to receive a reliable mean. The results of the simulation are used as the individual‘s fitness value.

Unfortunately, the simulator does not support threading for the parallel execution of simulation models. This design flaw was overcome by using ECJ’s master/slave architecture for performing the evaluations. This made it possible to speed up the evaluation steps substantially.

4.3 Webbased User Interface

The central management site runs on a Microsoft IIS Application Server and acts as the island server as well. Each island runs on a virtual machine with four processors each. The user interface (depicted in Figure 4) allows to select the participating islands and to set relevant parameters defining the heuristic’s behavior. It is possible as well, to select for each metaheuristic the evolutionary strategy. E.g. for GA between one- or two-point crossover and the associated mutation and crossover probabilities. When the optimization is started, the server sends the configuration via web service to the connected virtual machines. These start the islands according to the configuration, which then connect back to the main island server.

During the execution, the island prints information on optimization run to the command line. All occurring console outputs are intercepted and stored in the main database, running on the central management server. The user interface shows windows for each island in which the console outputs are shown. These are updated steadily to be able to observe the optimization process. All runs are stored with their parameters in the database for later analysis and evaluation.

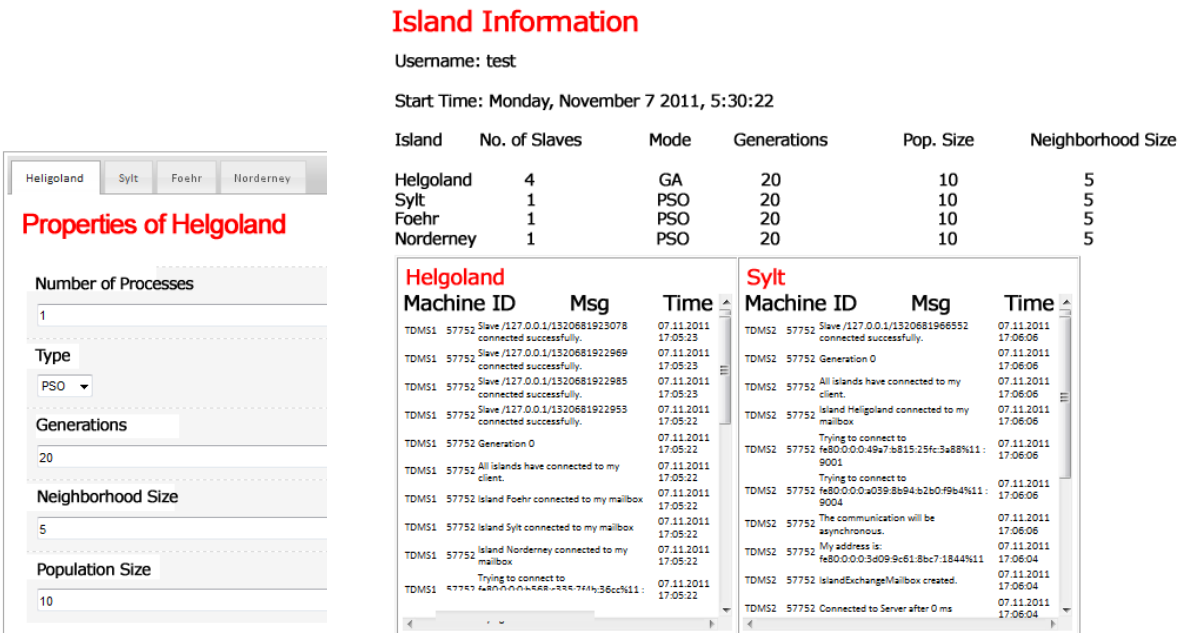

Figure 4: User Interface: Parameter setting for one island (left) and global optimization run output observation and analysis (right)

5 RESULTS

5.1 Evaluation Model

To evaluate our solution, we use a simple material flow simulation problem. The use case examines various possible layouts of a production system in planning for a small automotive supplier. The basic material flow simulation model is enriched by specific model features, allowing the dynamic calculation of total costs parallel to the performance evaluation of the material flow. Therefore, the results of a specific parameter configuration can, later on, be evaluated on a cost base. The goal is to set up a production system that fulfills capacity restrictions and minimizes cost (total cost for a specific parameter configuration is major KPI for the evaluation of a simulation run). Some parts of the system are fixed and some can be changed. For three workstations, the to be bought machine must be chosen out of several available machine types. These differ in total costs, machine hour rate and cycle time and maintenance frequency. The optimal cycle time of four machines must be determined to fit in the production system. The total output of work pieces of the system is restricted by a lower bound and upper bound. Violations to this restrictions are penalized with higher costs. To determine the optimal machine selection and configuration with minimal total costs at an given output level is the goal of the optimization study. It is a deterministic model in this proof of concept approach, so that no replications in this comparison are implemented.

5.2 Performance Evaluation

We tested our model on various combinations of islands and metaheuristics to evaluate our approach. Each run was performed with an population size for each island of 10, 20 and 30 individuals. Every island used four slaves for the evaluation step. Figures 5-7 are showing the development of the best found solution over 50 generations for selected island/metaheuristic combinations, measured in total cost as the KPI given as a result by the simulation model.

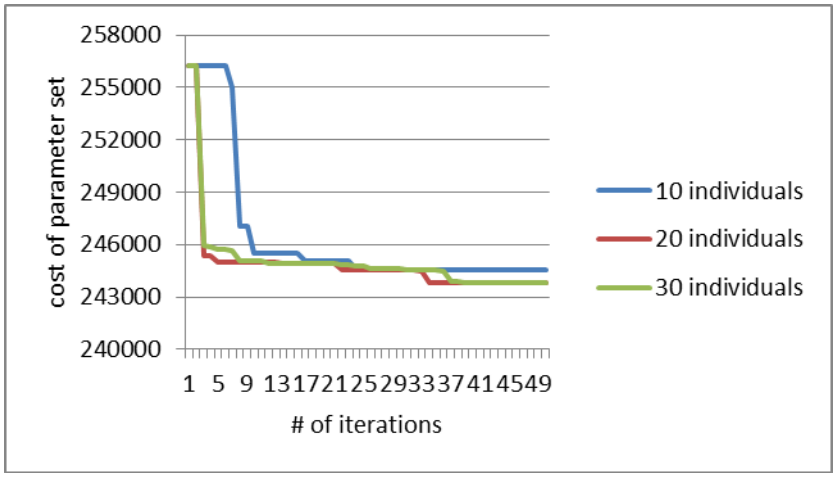

Figure 5: Performance of using PSO on a single Island

In a direct comparison for the single island approach, the Particle Swarm Optimization (Figure 5) converges considerably faster than the Genetic Algorithm (Figure 6). In all cases, feasible solutions without penalty costs are found already in the first generation.

The results of the combination of GA and PSO (Figure 7) with the exchange of individuals are comparable to the single island PSO case. Good solutions are found very early during the optimization and the overall best solutions are better as well. It is observable that the combination of Genetic Algorithm and Particle Swarm Optimization improves the results, even though the results of the metaheuristics on their own are inferior.

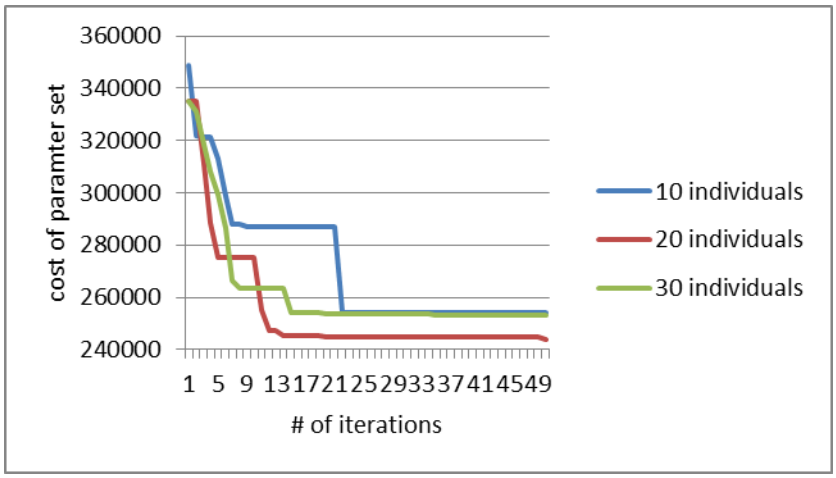

Figure 6: Performance of using GA on a single Island

With 20 and 30 individuals per island, near-optimal solutions can be found very quickly. Whereas with only 10 individuals, several generations more are needed to find equally good solutions. But since the evaluation of one individual takes a substantial amount of time, the total time to reach a good solution does not differ much. In this case a balance between the number of generations and the total time to find a near-optimal solution must be made.

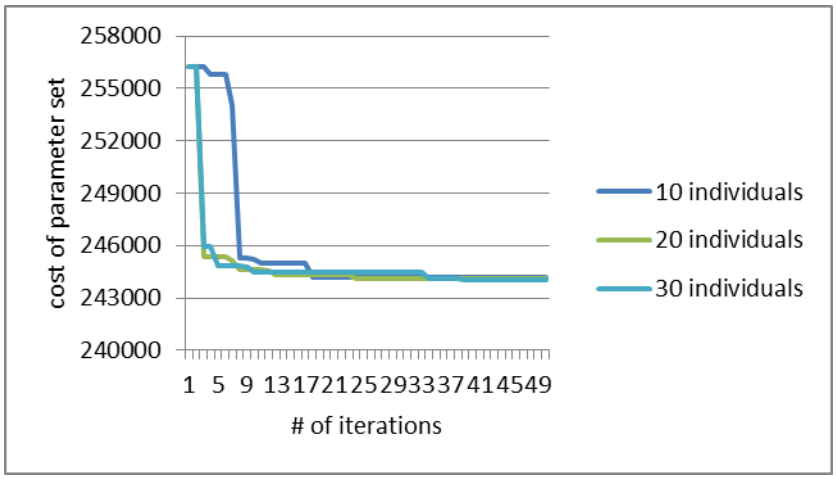

Figure 7: Performance of a combination of using GA and PSO on one island each

6 CONCLUSION & OUTLOOK

This paper presents an combinational approach of material flow simulation and meta-heuristic for an innovative, fast converging procedure for the optimization of simulation model's parameters. Based on a specific test setting, Particle Swarm Optimization, Genetic Algorithm as well as their combination are used as an automatic experiment design in a distributed simulation environment. The given results show in a first step, that the procedure in sum is converging fast and leads to optimal parameter values for the simulation model. Best performance is achieved by the use of Particle Swarm Optimization and Genetic Algorithm which transfer the best-found-solution from time-to-time during optimization.

Future work will be derived in the following areas: a) select, implement and test other heuristics, preferably from the area of evolutionary algorithms; b) improve further convergence of the overall procedure by stopping simulation scenarios with a given parameter set during execution, which is significantly worse and the gap to the best solution cannot be caught up.

ACKNOWLEDGMENTS

This work was significantly supported and influenced by the student's group work within the master course 'techniques of material flow simulation' in summer 2011 at the Heinz Nixdorf Institute of the University of Paderborn, Germany. Many thanks to Kevin Christ, Jan-Hendrik Fischer, Marta Hartmann, Florian Isenberg, Mathis Kuntze, Paul Markwart, Tom Oswald and Thomas Seebothe.

REFERENCES

Amdahl, G. M. 1967. "Validity of the single processor approach to achieving large scale computing capabilities" In *AFIPS Conference Proceedings*. Vol. 30. AFIPS Press, Reston, Va., 1967, pp. 483-485.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2000. *"Discrete-Event System Simulation"*. 3rd ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Blum, C. 2003. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison" In *ACM Computing Surveys (CSUR)*, 35(3), 268-308.

Cung, V., Martins, S., & Ribeiro, C. 2002. "Strategies for the parallel implementation of metaheuristics" In *Essays and Surveys in Metaheuristics*.

Dangelmaier, W., D. Huber, C. Laroque, and B. Mueck. 2005. "d^{3}FACT insight - An immersive material flow simulator with multi-user support". In *Proceedings of the 2005 Summer Computer Simulation Conference*, edited by A. Bruzzone and E. Williams, 239–242.

Hachicha, W., A. Ammeri, F. Masmoudi, and H. Chachoub. 2010. "A comprehensive literature classification of simulation optimisation methods" In *International Conference on Multiple Objective Programming and Goal Programming* - MOPGP10 No. May 24- 26, 2010. Sousse, Tunisia.

Kabirian, A., and S. Olafsson. 2007. "Allocation of simulation runs for simulation optimization," *Proceedings of the Winter Simulation Conference*, Edited by Henderson, S. G.; Biller, B.; Hsieh, M.-H.; Shortle, J. ; Tew, J. D. and Barton, R. R. ; pp.363-371, 9-12 Dec. 2007.

Krug W. 2002. "Coupling ISSOP with other simulation systems. Modelling, simulation and optimization for manufacturing, organisational and logistical processes", SCS European Publishing House, Erlangen, S 132–185.

Krug, W.; Wiedemann, Th; Liebelt, J.; Baumbach, B. 2002. "Simulation and Optmizationi n Manufacturing, Organisation and Logistics" in: *Simulation in Industry – Modeling, Simulation and Optimization – Proceedings of the 14th European Simulation Symposium* ,Edited by: Verbraeck, Alexander and Krig, Wilfried, SCS Press.

Kennedy J., R.C. Eberhart. 1995. *Swarm Intelligence*. San Francisco, Morgan Kaufmann Publishers Inc.

Laroque, C. 2007. *Ein mehrbenutzerfähiges Werkzeug zur Modellierung und richtungsoffenen Simulation von wahlweise funktions- und objektorientiert gegliederten Fertigungssystemen*. („A multi user tool for modelling and open ended simulation of optionally functional or object oriented structured production systems") HNI Verlagsschriftenreihe. Paderborn.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling & Analysis*. 3rd ed. New York: McGraw-Hill, Inc.

Luke, S. 2011. ECJ 2.0 - A Java-based Evolutionary Computation Research System. Accessed April 15. http://cs.gmu.edu/~eclab/projects/ecj/.

Fu, M. C. 2002. "Feature Article: Optimization for simulation: Theory vs. Practice". In *INFORMS Journal on Computing*. 14(3): 192-215.

Paul, R. J., and T. S. Chanev. 1998. "Simulation optimisation using a genetic algorithm" In *Simulation Practice and Theory*, Volume 6, Issue 6, 15 September 1998, Pages 601-611.

Rardin, R. L. 1997. *Optimization in Operations Research*. Prentice Hall, 1997.

Russell, S., and P. Norvig. 2009. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 2009.

Talbi, E.-G. 2009. *Metaheuristics – From Design to Implementation*. Hoboken, NJ: John Wiley & Sons.

AUTHOR BIOGRAPHIES

CHRISTOPH LAROQUE studied business computing at the University of Paderborn, Germany. From 2003 to 2007 he has been a Ph.D. student at the graduate school of dynamic intelligent systems and, in 2007, received his Ph.D for his work on multi-user simulation. Until September 2011 he was team leader of the "simulation & digital factory" at the chair of Business Computing, esp. CIM. Since October 2011, he is the temporary chair holder for modeling and simulation at the institute of applied computer science at the Technical University of Dresden. He is mainly interested in the simulation-based decision support for operational production and logistic processes. His email address is christoph.laroque@tu-dresden.de.

ALEXANDER KLAAS studied computer science at the University of Paderborn, Germany. Since 2010 he is a research assistant at the Heinz Nixdorf Institute. His research interests are knowledge based methods, AGV control algorithms, online optimization methods and discrete event simulations. His email address is Alexander.Klaas@hni.upb.de.

JAN HENDRIK FISCHER studies management information systems at the University of Paderborn, Germany. He has been pursuing his Master's degree since 2010 and is currently completing his thesis. The application of simulation-based optimization using evolutionary algorithms on production and logistic systems is his main research interest. His email address is Jan-Hendrik.Fischer@hni.upb.de.

MATHIS KUNTZE studies business informatics at the University of Paderborn, Germany. He has been pursuing his Master's degree since 2009 and is currently completing his thesis. Beside methods of simulation and optimization, he is also interested in parallel computing and computer networks. His email address is mathisk@mail.upb.de.