An Automatic Approach for Parameter Optimization of Material Flow Simulation Models based on Particle Swarm Optimization

Christoph Laroque
Institute of Applied Computer Science
Technical University of Dresden
Dresden, Germany
christoph.laroque@tu-dresden.de

Jan-Patrick Pater
Heinz Nixdorf Institute
University of Paderborn
Paderborn, Germany
ppater@hni.upb.de

***Abstract* - The analysis of production systems by the use of discrete, event-based simulation is widely used and accepted as decision support method. It aims either at the comparison of competitive designs or the identification of a "best possible" configuration of the simulation model. Here, combinatorial techniques of simulation and optimization methods support the user in finding optimal solutions, which typically result in long computation times and though often prohibit a practical application in today's industry. This paper presents a fast converging procedure as a combination of a swarm heuristic, namely the particle swarm optimization, and the material flow simulation to close this gap. Faster convergence is realized by a specific extension of classic PSO implementations. First results show the applicability with a simulation reference model.**

Keywords-***Design of Experiments; DES; meta-heuristics; particle swarm optimization; parameter optimization.***

I. 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 designed or improved, are typically projected to mathematical models and then optimized by the use of simulation and/or optimization technologies. In both disciplines, models as an abstraction of the real-world system are used to improve decision variables and resulting key performance indicators under a given set of restrictions, e.g., the identification of the maximum throughput of a production site or network. In simulation, this improvement is usually achieved by the iterative evaluation of multiple scenarios and their subsequent simulation results. In the case of optimization, the optimal configuration is achieved by mathematical optimization algorithms or (meta-) heuristic approaches [12].

Due to the high computational demand of these methods, specific procedures as a combination of both simulation and optimization were derived, to combine both advantages: an optimization algorithm can be used to automatically generate a specific model configuration, which can be evaluated by simulation runs [9]. 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 [9].

This paper presents a feasibility study for a specific combination of simulation and optimization, where material flow simulation for production processes is combined with the meta-heuristic approach of particle swarm optimization. The goal is the development of a fast converging procedure model, which can be applied in a practical, industrial environment. Especially in this area, the given complexity of the underlying production system, and thereby the simulation model, is very high, so that the application of standard combinatorial approaches of mathematical optimization and material flow simulation is prohibited, since it needs an excessive amount of computational power.

The paper presents in the following sections in short the necessary state-of-the-art in discrete, event-based simulation, methods for the design of experiments as well as particle swarm optimization. The conceptual approach of the procedure is presented in Section 3, followed by the prototypical implementation in the material flow simulation tool d³FACT. The first evaluation results of the procedure are shown in Section 5. The paper closes with an outlook on future work in this area.

II. STATE-OF-THE-ART

A. Discrete, event-based material flow simulation

Simulation, especially material flow simulation is a methodology in operations research, which uses a model to describe a real world system, which in turn is a collection of entities, interacting together [9]. It has some distinct advantages, since it enables the observation of a system's behavior even before it exists in reality and furthermore for very complex, dynamic models. Based on the generated insights, users are able to validate the design of the system, to develop strategies for its operation or to determine optimal operating configurations. It is particularly suitable to determine how the system can be configured in the best way, especially when it is difficult or very costly to change an actual system design or to test control rules for a specific material flow in real. Time periods can be reviewed much quicker, sensitivity analysis and a graphical representation of the simulation model and its dynamics are further

 ISBN: 978-1-61208-234-9

advantages. Consequently, this will apply, where analytical methods fail due to system's complexity and if there is no other way to analyze an existing or designed system [6],[13].

B. *Design of Experiments*

Design of experiments (also DOE [11] or experimental design) refers to the use of statistical techniques to create an efficient, systematic set of controlled experiments for collecting data efficiently in order to estimate relationships between independent and dependent variables through measurement. DOE is used in engineering to design physical experiments to determine physical relationships (e.g., effect of pressure and temperature on yield in a manufacturing process). 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, fractional designs, Plackett-Burman-experiments as well as response-surface method (RSM) or evolutionary optimization (EVOP) are used [11].

As mentioned above, the combination of simulation and optimization methods is also used and known, but today leads typically to high computational demands, which the practical application. Key factor for a successful application of such combinatorial approach is the fast convergence of the designed procedure.

C. *Particle swarm optimization*

Optimization algorithms can be distinguished in two areas, namely exact and heuristic methods [12]. Exact methods like linear or dynamic programming are based on a mathematical model and deliver an optimal solution. A great drawback is, however, that these procedures cannot solve some problems in an acceptable time period. Furthermore, they may not deliver solutions at all. Heuristics or meta-heuristic approaches are able to find a feasible solution in a shorter period of time, but cannot guarantee optimality. The GAP, the difference between the solution and an optimal solution, may not be reached [12]. Heuristics are problem specific procedures and can be divided into heuristics for generating intial solutions and heuristics, improving a given solution. Meta-heuristics are more generalized procedures, which can be applied to a broader range of problem instances. Typical examples in this domain are Taboo-Search, Simulated Annealing, Iterative Locale Search as well as Evolutionary Algorithms, Ant Colony Optimization, Swarm algorithms and Neuronal Networks [12].

The particle swarm optimization approach (PSO) was initially formulated by Kennedy and Eberhart in 1995 [8]. It is an evolutionary algorithm with a fixed population out of the group of meta-heuristics. It utilizes a population of individuals which create a set of solutions. A member of this population is called particle. Each particle has a specific position within the solution space, spanned by the sum of restrictions. Its basic idea is based on the Boid model of Reynolds, Heppner and Grenander [7]. Here, a boid is an individual, that moves in the same direction as its neighbor (alignment), but also tries to move to the middle of the group (cohesion). At the same time the individual tries to maintain a minimum distance to the others (separation). This behavior is inspired by natural phenomena, well known e.g. by bird or fish flocks. The information exchange between the individuals implements a social behavior. Kennedy and Eberhart developed collision-free particles, that search an optimal fitness value, the so called 'cornfield vector' [8]. Each particle stores the best positions found by the entire group as well as its personal best position

III. IDEA

A. *Fundamental approach*

The combined procedure of simulation and meta-heuristic presented in this section differs in an essential way from existing PSO-approaches, since the evaluation of the fitness of a particle is not possible in a direct way, but has to be derived through at least one, in most cases multiple simulation runs. Nevertheless, the core issue remains, that the simulation runs claims time and computing capacity and therefore a major objective to study is the determination of appropriate parameters for the algorithm, so that the number of required fitness evaluations is minimized.

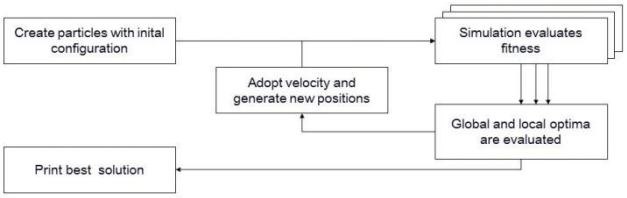

Figure 1. Principle procedure for parameter optimization.

Figure 1 shows the designed process of an automatic experimental design procedure as a combination of simulation and the PSO. Based on an initialization of the simulation model the input factors of the simulation model and their modeled limitations are derived. The PSO-algorithm then creates particles for a given number of configurations, which are to be evaluated in a first iteration. The simulation runs are fed by this configuration and evaluate the configuration by simulating and generating key performance indicators that allow the determination of the fitness value of each particle. The best solution found is stored and distributed to each particle and its position in the solution space, meaning the configuration of the simulation

 ISBN: 978-1-61208-234-9

parameters, is adjusted. Then, the next iteration for the simulation-based evaluation of the adjusted particles starts. The procedure repeats until a given termination criteria is reached (e.g., number of iterations, no improvement of the KPIs).

An important point of interest by using this approach is the application to complex systems with stochastic influences. This is due to their stochastic nature which always has a different result - and therefore a different fitness. A problem which arises here is the emergence of noise. One way of addressing this problem is evaluating a location several times to form an average (multisampling), or to exploit the history for sequential scanning. Here, the last values are smoothed to calculate a trend. The noise also causes particles not to reach the global optimum, as this cannot be precisely identified [2].

Additional improvements have been studied and introduced [3]. As the number of iterations of the algorithm is crucial for the computational effort, in conjunction with simulation, some modifications were made, that proved to be suitable in preliminary investigations. These were in detail:

- Intelligent positioning - The particles are specifically placed in certain areas of the solution space.
- Subdivision in three phases - exploration, exploitation and intensification. (further referred to as 3-phase PSO, cp. Section 3.2)
- Adaptive velocity - In the first phase (3 phase-PSO) the velocity of the particles is reduced to contain the explosion of the particles in the solution space.
- Selection - The particles of supposedly poor regions of the search space are relocated in good areas.
- Multisampling - The positions of the particles are evaluated several times and averages are calculated to reduce stochastic effects.
- Threshold - An additional stop criterion

B. *Construction of the 3-Phase-PSO algorithm*

The extensions to the PSO in the previous chapter provided some starting points. A compromise between exploring the search space and the convergence is to be found.

After an intelligent initialization the solution space can be subsequently investigated with a limited number of iterations and the adapted behavior of the particles. The next step is to select particles by their fitness values which are better than the rest. A further potential is obtained by exploiting a history for the previous positions as a particle could be pulled out by the global best position in the course of iterations from a good range. When only the current information is used, good areas might be neglected under certain circumstances. History allows that at the end of the first phase of the algorithm, the particles are repositioned in a well-defined number of top positions throughout the history. This selection has another positive side effect. The starting position of the particles has already been evaluated and is known. This again saves computational effort for the first iteration of the second phase. In total this adjustable first phase increases the diversification and the probability to find the global optimum [2].

In a second phase, the identified good areas have to be examined more closely, which is known as intensification in the domain of meta-heuristics. Here, the behavior of the particles is adapted to this stage. They are now trying to find the best possible solution in the subspace of the whole solution space that is being reinforced by an increased concentration of particles in this area. Furthermore, through formation of subpopulations reinforcement of behavior is achieved. The basic idea here is to examine defined regions of the search space in more detail and therefore not to be affected by disturbing influences of other areas. After a further, well chosen number of iterations, the areas can now be compared. Is the range on average substantially worse, it can be neglected; the parameterized essentials can be set by the decision maker.

As a part of the particles was filtered, the solution quality can be increased, with more iteration at the same computational effort as before. Because of the convergence properties of meta-heuristics few iteration at the beginning lead to some significant improvement in the objective value, while the last iterations achieve only marginal improvement. In the third and final phase, with a reduced number of particles, an improved result is searched as long as a predetermined number of iterations is not exceeded or improvement of a certain percentage level can be achieved.

IV. IMPLEMENTATION

A. *Standard PSO-algorithm*

The implementation of the PSO-algorithm was on a pre-implementation by Cingolani [5]. This is only an implementation of the initially described PSO algorithm and will be referred to as the 'Standard PSO'. The basic algorithm consists of a swarm S of particles x_i, and a fitness function $f(x_i)$. The swarm itself is a set of particles with a position vector

$$\vec{x}_i = (x_{1i}, x_{2i}, ..., x_{ni}),$$

where i denotes particle x_i and x_1 the index of the vector component. The velocity

$$\vec{v}_i = (v_{1i}, v_{2i}, ..., v_{ni})$$

is defined equally. The particles also consist of a memory of their personal best found position p_i so far as well as the current fitness value. The bound for every component of the position form the search space and all positions within are valid solutions, thus the solution space. For the swarm however, the global best found position p_g as well as the

global best found fitness are stored together. The position represents an allocation of numbers, which serves as input variables for the fitness function (here: the specific simulation model to be evaluated). The result of the fitness function is the fitness of the particle (here: the resulting simulation models KPIs). At the beginning of the optimization run, the positions are assigned randomly within the solution space. Then all positions are evaluated. The difference between the actual and the best position within the swarm results in a direction – the velocity. With this velocity a new position is calculated and the first iteration finishes. The next iterations form a cycle of position evaluation, velocity calculation and repositioning the particles. This is executed until stopping criteria is fulfilled.

In every iteration the global best position is evaluated which is formally defined as follows (here for a minimization, for maximization it would be *arg max*{…}):

$$p_g^k = \left\{ \{p_1^k, ..., p_n^k\} \middle| p_i^k = \arg\min\left\{ f(p_1^k), ..., f(p_n^k) \right\} \right\}$$

The velocity is calculated per vector component:

$$v_{id}^{k+1} = v_{id}^k + c_1 r_1 (p_{id} - x_{id}^k) + c_2 r_2 (p_{gd} - x_{id}^k),$$

where r_1 and r_2 are random numbers within an interval [0 ; 1], which results in a randomness of the particles movement. They can be calculated once and be applied as a scalar to the vector or calculated per component. c_1 and c_2 are called acceleration coefficients and as a result from their concrete assignment, it is determined whether the particle is attracted stronger to his personal or to the global best position.

$c_1 r_1 (p_{id} - x_{id}^k)$

is also called the *cognitive* and

$c_2 r_2 (p_{gd} - x_{id}^k)$

the *social* component. If the velocity is also set at initialization it can be restricted to keep within the solution space after the first iteration.

With a new velocity vector the positions can be refreshed as follows:

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1}.$$

Since the particles move free through the solution space it must be taken into consideration, that recalculated positions could be out of range. Most optimization problems are defined by restrictions, which are predetermined. A possibility to resolve these cases are so called walls (dimension limits). When the particles encounter the dimension limits, multiple methods exist, to set the position of the particle back within the solution space. Three methods are common and have been implemented [14]:

- Absorbing Walls: When a particle hits one of the dimensions limits, the speed in this dimension is set to 0. This creates the possibility that particles are set back into the solution space.
- Reflecting Walls: When a particle comes up against one of the limits, the algebraic sign of the velocity component in this dimension is inverted. The particle is set back in the direction of the solution space.
- Invisible walls: In this method, the particles can exceed the dimension limits. Particles that are outside these limits will not be evaluated. The motivation of this method is to save computational effort

Another common extension is *inertia weight* formulated by Shi and Eberhart [15]. Inertia weight is a factor and multiplied with previous speed of a particle v_{id}^k to contain explosion of the swarm, meaning that the velocity rises without limit. Inertia weight is by implication meant to accelerate the convergence of the swarm. An extended velocity formula results as follows:

$$v_{id}^{k+1} = \omega v_{id}^k + c_1 r_1 (p_{id} - x_{id}^k) + c_2 r_2 (p_{gd} - x_{id}^k)$$

Shi and Eberhart investigated ω within the interval [0 ; 1.4]. They revealed that within [0.8 ; 1.2] a quicker convergence can be achieved, while higher values (>1.2) result in a failure to converge. To small values can cause the swarm to get stuck in local optima.

Angeline [1] noticed that the classic PSO algorithm had an implicit weak *selection* of particles regarding the personal best fitness. The purpose of selection aims at placing particles in supposedly good regions of the solution space discovered earlier. Angeline proposed a method to perform an explicit selection:

- For each particle in the swarm: compare the fitness value with those of the others – each time the current particle has a better fitness it is rewarded.
- Sort particles ascending by number of rewards.
- Choose upper half of particles and copy their position to those from the lower half, while preserving the personal best position of each particle copied.

This procedure has to be performed before calculating new velocity vectors. Evangeline revealed that solution quality within local optima increased, while weakening the ability to find global optima. Consequently this extension improves the convergence of the PSO

Finding global optima requires the procedure to maintain diversity. Suresh et al. [16] proposed two modifications. One is about adjusting the inertia weight depending of the distance current global best position. They formulated it as follows:

$$\omega = \omega_0 \left(1 - \frac{\delta_i}{\delta_{\max}} \right) \quad \text{with } \omega_0 = rand(0{,}5\,;1),$$

with δ_i denoting the Euclidian distance in a *d*-dimensional space and $\delta_{\max}$ the greatest possible distance:

$$\delta_i = \sqrt{\sum_{d=1}^{D} \left(p_{gd} - x_{id}^k \right)^2}.$$

 ISBN: 978-1-61208-234-9

As a result Suresh et al. revealed that this adaptive adjustment substantially improved performance. As being intensively investigated, these extensions, modifications and improvements have been proposed to the standard PSO.

To address the issue of random influences in the fitness function we used multisampling as an further extension. This means that the same position is evaluated multiple times and an average fitness value is returned. The work of Bartz-Beielstein et al. [2] was a foundation. Using history data and smoothening of fitness values over a specified amount of iterations, inspired to abuse this feature in combination with the selection extension described earlier.

B. 3-Phase PSO implementation

According to the conceptual design, the implementation of the PSO algorithm was refined by the realization of the 3-Phase PSO according to the following 3 phases:

Phase I:

1. Initialization of *n* particles using intelligent positioning,
2. Initialization of the velocity vectors as *null*-vectors,
3. Evaluate fitness of each particle using distributed simulation runs and update velocity vectors with a weak attraction towards the global best position,
4. Save fitness values in history data,
5. Generate new position for each particle,
6. Repeat step 3-5 till termination criteria achieved.

Phase II:

1. Select *n* best found fitness values from history data and set particles to these positions,
2. Initialize velocity vector, aiming at the best found fitness for each group,
3. Evaluate fitness and update history data,
4. Update velocity vector,
5. Repeat 2-4 as long as improvement is gained.

Phase III:

1. Identify best group of particles,
2. Delete other particles,
3. Initialize velocity vector, aiming at the overall best found position,
4. Evaluate fitness and update velocity vectors,
5. Repeat, until criteria for termination are realized.

The selection feature presented in [1] was investigated with the result that the 3-Phase-PSO was stuck in a local optimum. The intelligent initialization with the greatest possible dispersion strongly reduced this risk. Here, the particles are scattered evenly in the feasible solution space so that the swarm of particles is provided with a rough overview. An original random initialization of the starting positions could lead to the fact that all particles concentrate only in a certain region of the solution space. The remaining solution space would not be evaluated. Selection in conjunction with an intelligent initialization and an adaptive swarm behavior can strongly reduce the number of fitness evaluations. The objective of the fast converging procedure needs to quickly identify good areas and to concentrate the particles in these regions and to achieve good improvements in fitness in as few iterations as possible, without limiting the global search.

C. Material flow 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 [6],[13].

For a first evaluation of the feasibility study a rather simple simulation model was selected, which is presented in Figure 2. Reason for such a selection was to crosscheck the conformability of the generated solutions by the designed procedure with a simple reference model. The resulting parameter configurations could easily be evaluated by just logical thinking.

Another advantage of this simple reference simulation model was a significant lower amount of necessary simulation time for each evaluation. Since all developed improvements of the standard PSO-algorithm, the 3-Phase-PSO and overall procedure had to be evaluated by multiple simulation runs, the evaluation could be limited in time.

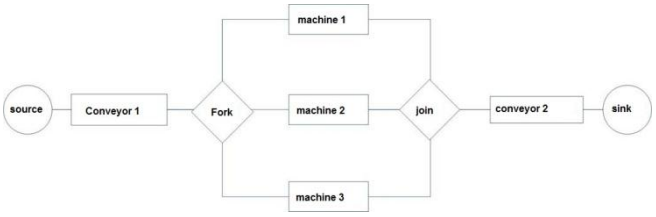

Figure 2. Simple material flow reference model in d³fact.

D. Graphical user interface

For a more user-friendly evaluation of the different configuration parameters of the PSO as well as the 3-Phase-PSO algorithm, a simple, graphical user interface was developed. Based on the initial simulation model, the parameter limits for the simulation configuration set as well as the PSO algorithms can be configured. Based on that, out of the GUI, the overall procedure process is started. The application starts the simulation runs with each particle's simulation configuration and shows the simulation results. Figure 3 shows a screenshot of the graphical user interface for the procedure testing.

 ISBN: 978-1-61208-234-9

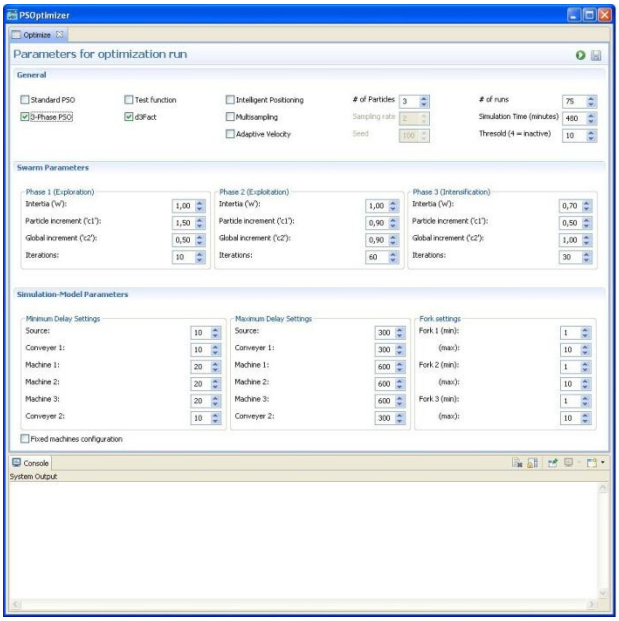

Figure 3. Screenshot of the user interface PSOptmizer.

V. RESULTS

To ensure the correct operation of the PSO, a mathematical test function has been used initially. The functions global optimum is predictable by analytical methods. It also has multiple local optima. The evaluation of the simple simulation model was realized in d³FACT. The material flow network is built at runtime with a model configuration, which corresponds to the starting position of the particle in the solution space. The solution space is defined on the GUI in the ranges of model parameters. After establishing the model, it is simulated for the set duration and the material flow is returned as fitness to the PSO.

The test runs were conducted on several computers with two core processors and four gigabytes of memory. There were as many instances of the PSOptimizer started as cores available. These were assigned to a core to reduce losses through the operating system scheduler. Note that it could still come to computational losses due to simultaneous access to the working main memory. Run time played a minor role in these first investigations and therefore such losses have been neglected. Early trials showed, that there can be significant variations in the best objective value test run. This was caused by the stochastic factors in the model as well as in the PSO. So, each configuration was calculated 200-times to get a sample, which then was evaluated by statistical methods. The arithmetic mean for the iterations of the algorithm was calculated as the best achieved objective function values. For this, the GAP could be determined.

A. *Mathematical test function*

Besides demonstrating the feasibility of the designed procedure, a first investigated objective was to evaluate the PSO-extensions according to the reduction of necessary iterations, especially in conjunction with the simulation environment. Here, the focus was set on the PSO algorithm so that a minimum possible number of iterations was necessary for the best possible solution. In order to allow a fast evaluation of the PSO-algorithm configuration sets, a simple mathematical function was used:

$$f(x,y) = \sin(x) + \sin(y) + 0.1x + 0.2y$$

Table 1 shows some resulting GAP-values of the implemented PSO-algorithms according to the test function:

TABLE I. RESULTING GAP FOR MATHEMATICAL TEST FUNCTION

Error for test function				
particle	**standard PSO**	**standard PSO (int. position.)**	**3-Phase PSO**	**3-Phase PSO (int. position)**
6	4,64%	0,65%	2,39%	0,25%
9	2,25%	0,42%	1,44%	0,12%
11	1,03%	0,35%	0,42%	0,07%
22	0,30%	0,10%	0,10%	0,03%
33	0,26%	0,06%	0,01%	0,01%
GAP (mean)	**1,70%**	**0,32%**	**0,87%**	**0,10%**

The most important and up to date new extension was the intelligent positioning of a set of particles in the solution space. This option significantly increased solution quality and nearly zeroed the possibility of getting stuck in a local optimum. Figure 4 displays the impact of this feature in iteration counts.

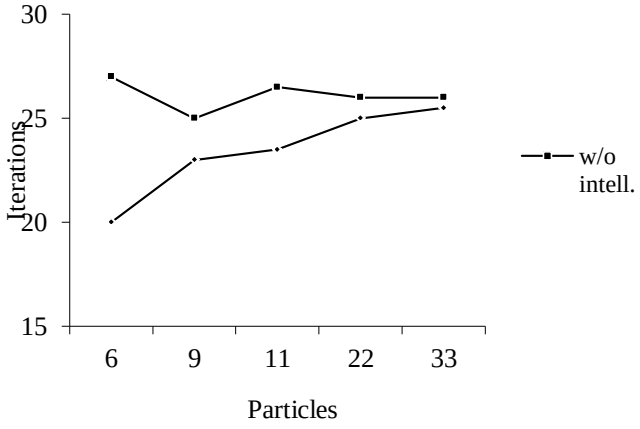

Figure 4. Iteration count for Standard PSO on test function.

B. *Reference simulation model*

The model was deliberately kept simple to determine the theoretical maximum throughput by analytical means. This eased an investigation of GAPs. The design of the experiment and the data obtained were analyzed, processed and are presented below. In addition to the preset maximum

iteration number, a threshold as a criterion was selected. For the analysis presented below the threshold was set at ten.

Table 2 shows some resulting GAP-values of the implemented PSO-algorithms according to the simulation model in d³fact.

TABLE II. RESULTING GAP FOR D³FACT SIMULATION MODEL

Error for simulation particle	standard PSO	standard PSO (int. position.)	3-Phase PSO	3-Phase PSO (int. position)
6	10,47%	9,00%	8,80%	4,79%
9	7,27%	3,51%	5,75%	0,77%
11	6,85%	0,60%	3,86%	0,75%
22	2,61%	0,26%	1,42%	0,07%
33	5,70%	2,82%	4,11%	1,29%
GAP (mean)	**6,58%**	**3,24%**	**4,79%**	**1,54%**

C. Conclusion of results

Figure 4 demonstrates that the swarm parameters chosen for every phase are far from best in term of iteration count. Considering the GAP values listed in table 1 and table 2 the goal has been fulfilled. In terms of solution quality, it has to be considered that the random input in the swarm results in random behavior of the particles. In one run the particles may have more "luck" than the next. In conjunction with simulation based optimization such a case is inconceivable. Since the possibility of such a result is nearly zeroed and result quality can be raised at least without increase in total runtime the presented options still perform quite well.

However, not all proposed extensions lead to the desired reduction in iteration count. The adaptive velocity option seemed to slow down the algorithm especially in when the number of particles is low. It turns out that the PSO can manage the velocity of the particles better by itself than when interfering with a bound.

Since the swarm parameters were chosen with respect to the quality of the solution and a parameter study for those was not conducted at this point, potentials for improvement may left open. Another reason for this result is that the swarm parameters in the distinct phases were set to values that are boundaries of internals that have been investigated by researchers earlier [15].

Due to model characteristics the difference between both algorithms is not big as depicted in Figure 5. The models optimal solution lies at the fringe of the solution space. The implemented restriction violation procedure fosters the identification of solutions in these regions.

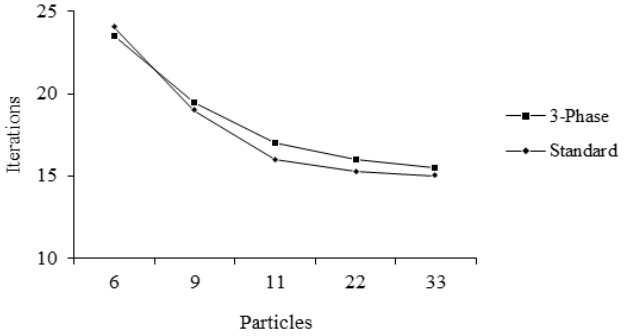

Figure 5. Iteration count for 3-Phase vs. Standard PSO with d³fact.

Not all modifications introduced had the desired success. The test runs, however, show a first conclusion: the configuration of the swarm is not arbitrary. Parameters such as population size and the choice of the threshold have to be done carefully. Population size is a key success factor for the necessary effort and solution quality. A higher threshold does not significantly increase the solution on average, but the effort is considerably. The investigated threshold was 10 and 20. The difference between the GAPs for all standard test runs of these two values is only 0.17%, while the average duration almost doubled.

The solution quality can be improved significantly by activation of intelligent positioning (cp. Figure 6). The studies show that the particle number and dimensionality may cause weaknesses. This plays a particular role as the last two particles are set. As a possible improvement, those two positions should always be occupied.

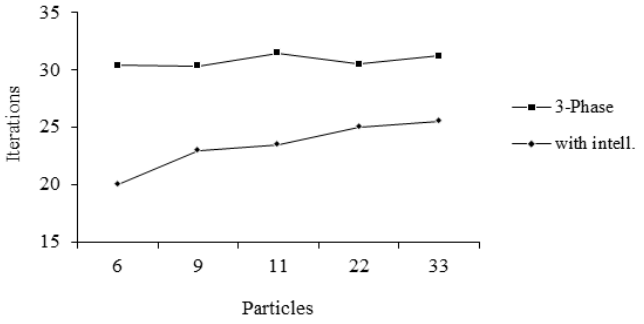

Figure 6. Iteration count for 3-Phase vs. Standard PSO with the test function.

The studies have also shown that the implemented selection in the 3-phase algorithm in conjunction with a history of positions leads to an improvement of GAP. It can also be shown that the question whether the selection should be performed several times remains open. Potentials for improvement exist. An investigation regarding the swarm parameters was impossible till today due to the complex

combination possibilities. There is need for further experimentation as the swarm parameters implemented in each respective phase are adapted according to the behavior of the swarm, but their quality remains unclear at this stage. Overall, the studies show that the 3-phase PSO-algorithm construction is a solid basis for further developments.

VI. OUTLOOK

With the implemented toolkit, it has been successfully shown that simulation models, in particular the configuration of parameters, can be optimized by a combination of simulation and the meta-heuristic particle swarm optimization. The test model differs in its complexity still far from real-life simulation models, since simulation itself is only useful for such complex models that deny an exact mathematical analysis. Nevertheless, as a proof of concept, the feasibility was shown in principle with the existing prototype.

Initial findings are collected on the behavior and the duration of the 3-phase PSO. It turns out that this first draft can be furthermore improved. Potential still exists in a more intensive use of selection and a better adaptation of the swarm parameters itself in the various phases; this would again improve the needed computational time. The solution's quality has already been significantly improved by the introduced modifications, described and presented in this paper. Further calculations and tests on the PSO-algorithms parameter setting are to be made in near future.

Another desirable future work was the evaluation of other heuristics as the optimization component in this approach, e.g., genetic algorithms. Even the combined approach of multiple heuristics for the optimization of the parameters could be taken into consideration.

ACKNOWLEDGEMENTS

This work was significantly influenced by the student work of Barthold Urban as well as the strong support of Markus Eberling, PhD-student at the chair of knowledge-based systems at the University of Paderborn.

REFERENCES

[1] P. Angeline, Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), Proceedings of evolutionary programming VII, 1998, pp. 601–610, Berlin, Springer.

[2] T. Bartz-Beielstein, D. Blum and, J. Branke, Particle swarm optimization and sequential sampling in noisy environments. In Metaheuristics International Conference, edited by R. Hartl and K. Doerner. 2005, pp. 89-94. University of Vienna.

[3] F. van den Bergh, An Analysis of Particle Swarm Optimizers. Ph.D. thesis, University of Pretoria, Pretoria, South Africa, 2002

[4] M. Clerc and J. Kennedy. The particle swarm - explosion, stability and convergence in multi-dimensional complex space. In IEEE Transactions on Evolutionary Computation. Volume 6. 2002, pp. 58-73, Washington, IEEE Computer Society.

[5] P. Cingolani, Open source particle swarm optimization library written in Java. Available via <http://jswarm-pso.sourceforge.net/> [accessed November 8, 2012].

[6] W. Dangelmaier and C. Laroque, Immersive 3D-Ablaufsimulation von richtungsoffenen Materialflussmodellen zur integrierten Planung und Absicherung von Fertigungssystemen. In Leobener Logistik Cases - Management komplexer Materialflüsse mittels Simulation. 2007, DUV Verlage.

[7] F. Heppner and U. Grenander, A stochastic nonlinear Model for coordinated Bird Flocks. In The Ubiquity of Chaos, edited by E. Krasner, 1990 pp. 233-238. AAAS Publications.

[8] J. Kennedy and R.C. Eberhart, Swarm Intelligence, San Francisco, Morgan Kaufmann Publishers Inc, 1995

[9] A.M. Law and W. D. Kelton, Simulation modeling & analysis. 3rd ed, New York: McGraw-Hill, Inc., 2000

[10] Leandro dos Santos Coelho, A quantum particle swarm optimizer with chaotic mutation operator, In: Chaos, Solitons & Fractals: Volume 37, Issue 5, 2006 pp. 1409-1418.

[11] D.C. Montgomery, Design and Analysis of Experiments, 7E International Student Version, John Wiley & Sons, Limited, 2008

[12] R.L. Rardin, Optimization in Operations Research, Prentice Hall, 1997

[13] H. Renken, C. Laroque and, M. Fischer, An Easy Extendable Modeling Framework for Discrete Event Simulation Models and their Visualization. In: Proceedings of The 25th European Simulation and Modelling Conference (ESM˙2011), 2011

[14] J Robinson and Y. Rahmat-Samii, Particle Swarm Optimization in Electromagnetics. In Journal of Artificial Evolution and Applications. 2008, doi:10.1109/TAP.2004.823969

[15] Y. Shi and R. C. Eberhart, A Modified Particle Swarm Optimizer. In Proceedings of World Congress on Computational Intelligence, 1998, pp. 69-73. Anchorage, IEEE Computer Society.

[16] K. Suresh, S. Ghosh, D. Kundu, A. Sen, S. Das, and A. Abraham, Inertia-Adaptive Particle Swarm Optimizer for Improved Global Search, In Proceedings of the 2008 Eighth International Conference on Intelligent Systems Design and Applications. Volume 2, 2008, pp. 253-258. Washington, IEEE Computer Society.

 ISBN: 978-1-61208-234-9