Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

COUPLING SIMULATION WITH HEURISTICLAB TO SOLVE FACILITY LAYOUT PROBLEMS

Andreas Beham
Monika Kofler
Stefan Wagner
Michael Affenzeller

Heuristic and Evolutionary Algorithms Laboratory
Upper Austria University of Applied Sciences
School of Informatics/Communications/Media
Hagenberg, 4232, AUSTRIA

ABSTRACT

In this paper we describe the optimization of a facility layout scenario, which involves coupling simulation with the optimization environment HeuristicLab. For this purpose we show a problem formulation that acts as an interface between these two domains of problem modeling and optimization, and discuss optimization methodologies and their results for a number of artificial test problems as well as more complex real-world problems. HeuristicLab was designed with both practitioners and algorithm developers in mind. Practitioners benefit from a graphical user interface that facilitates so-called interactive algorithm engineering, where algorithms can be adjusted without actually writing code. Algorithm developers are aided in the development process by the plug-in based, easily extensible architecture and integrated parallelization functionality.

1 INTRODUCTION

In the last decade a great deal of research has been devoted to a successful coupling of simulation models with optimization. To satisfy the ever increasing demand for simulation optimization from both researchers and practitioners, commercial simulation software is now frequently shipped with integrated or add-on optimization packages, such as for example AutoStat™(Carson 1997), OptQuest® (Sadowski and Bapat 1999), or the WITNESS® Optimizer (Rawles 1998). The listed commercial software solutions frequently employ metaheuristic algorithms for optimization, listing among others genetic algorithms, evolution strategy, tabu search, simulated annealing, scatter search, or hill-climbers. The vendors try to help the analyst to optimize simulation models (i.e., input parameters, structural assumptions) without burdening the user with too much algorithmic complexity. Therefore, the optimization interfaces usually expose only a limited set of algorithm parameters that can be tuned, and focus on statistical analyses of the optimization result rather than the characteristics of the optimization run, such as convergence behavior.

This black box approach is certainly favorable with respect to general-purpose robustness and usability of an embedded optimization tool but also limits the potential of the employed metaheuristics. We believe that advanced practitioners would profit from more transparency in simulation and optimization. In particular, the optimization environment should provide a diverse set of classical and advanced optimization algorithms to choose from and hide unnecessary complexity from the user who might not be an expert on metaheuristic optimization, while still providing ways to adapt and extend these algorithms on code-level if so desired. In this paper we show how the optimization framework HeuristicLab (Wagner 2009), which has recently been extended to communicate with simulation software such as AnyLogic™ or SimuLink®, addresses these issues. We also apply the software to optimize a facility layout with respect to material transport costs between production facilities.

The rest of the paper is organized as follows: Section 2 summarizes previous work on simulation optimization with HeuristicLab and gives a brief overview about the key features of the framework. In Section 3, a motivation for the facility layout problem and pending difficulties are provided. Moreover, the quadratic assignment problem (QAP) is introduced as a way to represent facility layout problems where the layout objective is to reduce material transport costs. The QAP is later refined to formulate the so-called machine placement problem (MPP), which is described in detail in Section 3.2. Section 4 focuses on metaheuristic optimization approaches and briefly introduces simulated annealing, genetic algorithms

and evolution strategies. These algorithms are applied to solve the MPP and results are described and analyzed in Section 5 and 6 respectively. Finally, Section 7 concludes the paper with a summary of the findings and an outlook on future work.

2 PREVIOUS WORK ON HEURISTICLAB AND SIMULATION

HeuristicLab is a framework for heuristic and evolutionary algorithms, developed and successively applied by members of the *Heuristic and Evolutionary Algorithms Laboratory* since 2002. Among other aspects, a sophisticated graphical user interface distinguishes HeuristicLab from existing heuristic optimization frameworks, which usually require comprehensive programming skills to adjust and extend the algorithms for a particular problem. In HeuristicLab algorithms are represented as operator graphs and changing or rearranging operators can be done by drag-and-drop without actually writing code (Wagner et al. 2008). The framework thereby shifts algorithm development capability from the software engineer to the user and practitioner. It is of course still possible to extend the framework on code level and algorithm developers benefit from HeuristicLab's light-weight plug-in mechanism that allows them to integrate custom algorithms, solution representations or optimization problems with ease. This concept leads to a significant level of code reuse across metaheuristic variants and encourages users to successively gain more knowledge about algorithm development (Wagner et al. 2007, Wagner 2009).

The optimization framework HeuristicLab has successfully been coupled with a variety of simulation frameworks to optimize selected input parameters of simulation models. Affenzeller et al. (2007) initially developed the conceptual design for simulation optimization with HeuristicLab. The actual implementation was done by Beham et al. (2008), who subsequently generated dispatching rules with genetic programming to optimize a simulation model of a semiconductor manufacturing plant. He also applied evolution strategies to optimize a supply chain model shipped with AnyLogic^{TM} 6 (Beham et al. 2008). Further tests regarding robustness in stochastic simulation-based optimization were conducted in Beham et al. (2009). Moreover, HeuristicLab has been used to optimize the image quality of a medical ultrasound transducer simulator (Kofler et al. 2008) and to solve a buffer allocation problem (Can, Heavey, and Beham 2008a, Can, Heavey, and Beham 2008b). The exchange of input and output data between HeuristicLab and the simulation model can be file-based, as employed for the ultrasound simulation optimization, or require a more sophisticated mechanism. For this purpose, a protocol layer has been implemented that allows the definition of a data exchange procedure via a state machine, thus allowing the injection or extraction of data from the simulation model at pre-defined synchronization points.

3 FACILITY LAYOUT PROBLEM

Layout planning for a production plant includes decisions regarding the physical allocation of work areas in a facility. The planner has to take various business strategic considerations and additional constraints into account, such as for example movement costs of resources (people, material, and machinery) between working areas, enforced placement and size restrictions, expected waiting times, required material buffer zones, and plant safety. The relevance of the listed constraints and target values is of course specific to the particular production scenario and must be defined together with plant experts. In this study we will focus on the material transport costs and assume an unconstrained, rectangular production floor.

It should also be noted that optimizing a facility layout once might not suffice, if the production scenario changes over time. In this case it could be necessary to adjust a carefully designed and calibrated layout to account for internal and external influences. For example changes in production volumes, work flow processes, technology, or product portfolio might call for a re-layout of sections or the whole production floor. A flexible layout should be either robust enough to tolerate such fluctuations or easily and cost-efficiently adjustable to reflect changing environmental conditions. For the real-world problems considered in this paper, re-layout is so far not a relevant issue, but the necessity could arise at a later stage of our project.

Within the operations research community numerous benchmark problems have been introduced that deal with certain aspects and objectives of facility layout planning. These include the quadratic assignment problem and various 2D cutting and packing problems such as bin packing or strip packing (Lodi, Martello, and Monaci 2002). Variants of the quadratic assignment problem are most commonly employed to model facility layout planning, if the core objective is the reduction of material transport costs.

3.1 Quadratic Assignment Problem

The quadratic assignment problem (QAP) was initially introduced by Koopmans and Beckmann (1957). The goal of the QAP is to assign a number of facilities to a number of locations in such a way that flow costs between the facilities are minimized. The problem is fully specified by a distance matrix for the locations and a flow matrix that represents the strength

of the coupling between pairs of facilities. Facilities can for example denote machines in a manufacturing plant or medical units in a hospital (Hahn and Krarup 2001). In these cases, the respective flow might be the number of material transported between machines or the number of patients that have to visit a sequence of units in the course of a medical intervention. In the classical formulation of the QAP the number of facilities equals the number of available locations.

The QAP is a combinatorial problem that has been shown to be NP-hard by Sahni and Gonzalez (1976) and is still challenging to solve even for modest problem sizes of $n > 25$ (Drezner, Hahn, and Taillard 2005). Common QAP benchmark problems were collected since 1991 in the QAPLIB (Burkard, Karisch, and Rendl 1997) and are also available online (Burkard et al. 2009). The QAPLIB maintainers also keep an up-to-date list of researchers working on the QAP and reference the algorithms that were successfully applied to various QAP problem instances, such as tabu search, simulated annealing, evolution strategies, ant colony optimization, scatter search, or GRASP.

3.2 Machine Placement Problem

3.2.1 Problem Formulation

We extended the classical formulation of the QAP to model the real-world scenario of planning a new factory more accurately. A global view of our scenario is given in Figure 1 which shows the involved domains and the problem formulation acting as an interface between them. In detail, we were interested in the optimal placement of a set of machines on a production floor to minimize material transport costs. Machines are represented as rectangles with given space requirements (x and y dimension) and can be placed in an arbitrary position on the floor. A buffer zone is reserved in the upper left corner of each machine, which is used to store and handle incoming or outgoing material. Moreover machines can be rotated by 90 degree steps, which changes the position of the buffer zone. The distance between two buffer zones is calculated as the Manhattan distance between their respective x and y coordinates. This problem is similar to the (dynamic) facility layout problem (D)FLP as for example given in McKendall, Shang, and Kuppusamy (2006), however in their work the machines or departments were of equal size and possible locations for their placement as well as the distance matrix were predefined.

The frequency of material flow between two machines is obtained from an external source and stored in a flow matrix. In a real production facility material flow between machines is hardly uniformly distributed. Data obtained from enterprise resource planning systems as well as simulation models provide different views on the occurring flows. A combination of these views results in flow matrices that can describe the real situation quite accurately.

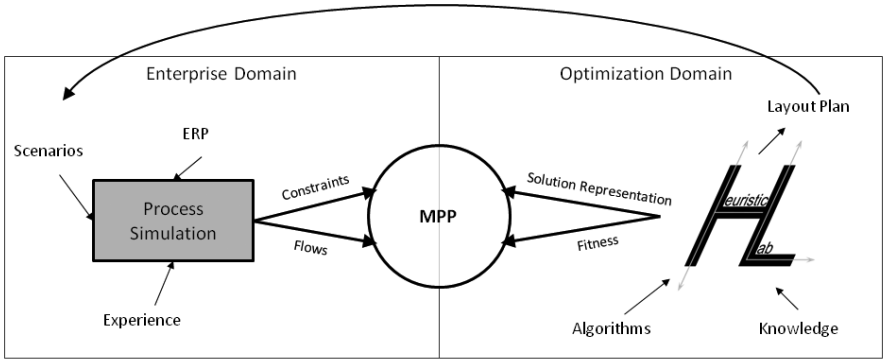

Figure 1: The interplay of process simulation and optimization on the MPP

3.2.2 Problem & Solution Representation

Problem, solution instance and objective function of the machine placement problem (MPP) are defined as follows:

- An MPP problem instance is fully specified by the x- and y-dimension of the floor, the dimensions of each separate machine, and the material flow matrix or matrices.
- A solution candidate of the MPP can be represented as three integer arrays, $\vec{x}$ and $\vec{y}$ for the center coordinates and $\vec{\omega}$ for the rotation states of the machines, respectively. Each machine can be in one of 4 rotation states, zero being the default, where the buffer zone is located in the upper left corner of the machine. States one to three denote

the remaining 90 degree counter-clockwise rotation steps. This representation allows the creation of unfeasible candidates, i.e., solutions with overlapping segments, a problem which is addressed in Section 3.2.3.

- Since material transport costs should be minimized, the objective function is defined as the total sum of all buffer zone distances weighted by the respective flow factor.

3.2.3 Dealing with Unfeasible Solutions

Metaheuristic algorithms are most directly applicable to unconstrained optimization. In case of constraints a way must be found to deal with unfeasible solutions. In the current formulation of the MPP, solutions with overlapping machine sections or sections that protrude the valid floor space can be generated. Common strategies to address this issue are discussed by Yeniay (2005), among them the discard or repair of unfeasible solutions and the employment of penalty functions. The most common issue with penalty functions is their correct parameterization, which is often problem-specific and obtained by trial and error search or costly meta optimization. These parameters heighten the complexity of algorithm tuning, which already involves a multitude of parameter decisions for each metaheuristic, such as the choice of suitable crossover and mutation operators, a population replacement strategy, population size, and mutation rate. For the test scenario in this paper we therefore use only a very basic additive penalty scheme, penalizing each constraint violation with a constant factor.

We denote the MPP solution representation vectors as $\vec{x}$, $\vec{y}$ and $\vec{\omega}$, the dimensions of the available production floor as x and y, and the flow and machine dimension matrices as F and D. Given the objective function $O(\vec{x},\vec{y},\vec{\omega},x,y,F,D)$, which returns the total sum of all buffer zone distances weighted by the respective flow factor, and the function $I(\vec{x},\vec{y},\vec{\omega},x,y,D)$, which returns the number of invalid, meaning overlapping or out of bounds, sections, we can now define the evaluation function

$$E(\vec{x},\vec{y},\vec{\omega},x,y,F,D) = O(\vec{x},\vec{y},\vec{\omega},x,y,F,D) + I(\vec{x},\vec{y},\vec{\omega},x,y,D) \cdot P(\vec{x},\vec{y},\vec{\omega},x,y,F), \tag{1}$$

$$\text{with a penalty factor } P(\vec{x},\vec{y},\vec{\omega},x,y,F) = (x+y) \cdot Max(F) \cdot Length(\vec{x}). \tag{2}$$

The penalty factor multiplies the maximum possible floor distance by the maximal flow and weights it with the number of machines. It thereby ensures that valid and invalid solutions occupy different ranges of the evaluation domain. This penalty scheme is not very sophisticated but circumvents the parameter tuning problem and turned out to be quite sufficient for the conducted experiments.

4 METAHEURISTIC ALGORITHMS

The no free lunch theorem (NFL) postulates that no given optimization strategy can be better than any other on all kinds of optimization problems. In other words, for each optimization strategy one can find a problem and according fitness landscape where the strategy will perform bad (Wolpert and Macready 1997). One consequence of the NFL is the development of a multitude of different optimization strategies, each with unique characteristics and the hope that it performs better than others on a certain subset of all optimization problems. However, testing a given problem with several different strategies is a complex task and requires considerable programming effort. Generic frameworks have evolved that offer an abstraction of optimization requirements in order to minimize programming effort and maximize reuse. HeuristicLab is such a framework which shifts the application of optimization strategies from an implementation point of view to a modeling point of view. In HeuristicLab algorithms are modeled by combining several generic parts using a graphical user interface. Similarly, problems are abstracted such that they make use of a certain representation and provide a fitness function as well as import parsers and graphical representations. The underlying representation, also called the encoding of a solution, provides manipulation operators such as crossover or mutation.

4.1 Simulated Annealing

Simulated annealing (SA) is among the oldest metaheuristics which offered an explicit strategy to escape from local optima. Its origins date back to the early days of scientific computing, when Metropolis et al. (1953) proposed an algorithm that can be used to efficiently simulate the thermal motion of atoms during a cooling process. Kirkpatrick, Gelatt, and Vecchi (1983) generalized the metropolis algorithm by replacing the atom's energy with a cost function, termed it *simulated annealing* in analogy to the annealing process in metallurgy, and successfully applied the algorithm to optimize computer chip design and the traveling salesman problem. SA employs the temperature as a simple control parameter that guides the search and balances phases of diversification and intensification. As the algorithm proceeds, a step-wise reduction of the temperature

focuses the search on a promising region of the solution space, which eventually leads to convergence. The algorithm's performance depends largely on the initial temperature as well as on the cooling scheme. If the temperature drops too quickly the search might get stuck in a worse local optimum; on the other hand, if the cooling is too slow, the algorithm might not have converged when it reaches the stop criterion, which might be for example a maximum number of evaluated solutions. A typical annealing scheme is multiplicative annealing which decreases the temperature in an exponentially shaped curve. Figure 2 displays a schematic chart of the simulated annealing algorithm. In our case an inner loop is performing neighborhood sampling at a given temperature until either an accepting solution is found or a maximum effort reached.

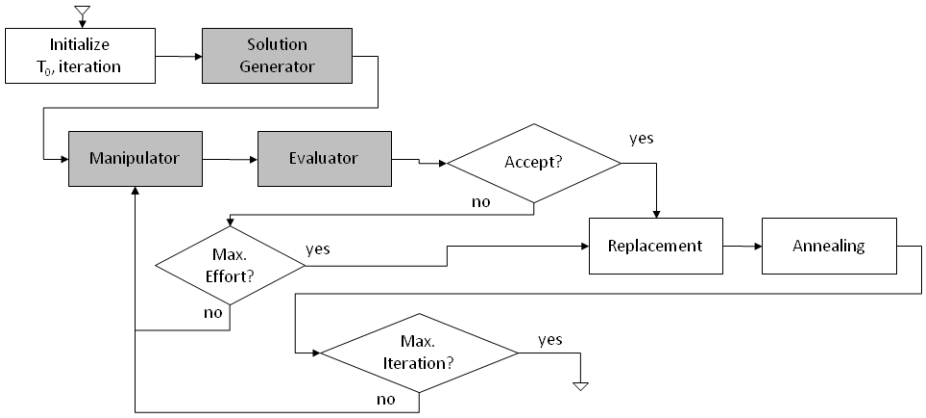

Figure 2: Schematic display of simulated annealing, the gray boxes mark problem-specific operators

4.2 Genetic Algorithms

Genetic algorithms (GAs) were introduced in the early 70s by John Holland as what is now known as the canonical genetic algorithm using binary solution encoding, a fitness-proportional selection, single point crossover, and bit flip mutation. John Holland proved that in this configuration so-called building blocks evolve over time which in turn are assembled in individual solutions such that finally a reasonably good solution is achieved (Holland 1975, Goldberg 1989). Over the years the GA community has expanded the application of genetic algorithms to other domains and introduced additional encoding schemes as well as crossover and mutation operators. Empirically obtained results show that GAs are able to perform well under such conditions (Affenzeller et al. 2009, Doerner et al. 2007).

The genetic algorithm used in this study is based on what has been established as the standard genetic algorithm (SGA) (Dumitrescu et al. 2000) that applies crossover to all individuals, mutates them with a certain, and generally low, probability, and replaces the whole parent generation with the new children except for a small number of elite parent individuals that are allowed to survive.

4.3 Evolution Strategies

At about the same time genetic algorithms were developed, evolution strategies (ES) appeared in first publications. Originally, they differed from the genetic algorithm such that the main operator responsible for the creation of solutions is the mutation operator. Ingo Rechenberg and Hans-Paul Schwefel introduced several basic variants of ES before adopting more complex solution manipulation concepts, such as recombination, to broaden the applicability of the evolution strategy (Rechenberg 1973). The first variant is the simple (1+1)-ES where a mutant was created from a given parent which replaced the parent only if it was better. It was shown that such a simple strategy coupled with a more intelligent adaption of the mutation strength allowed the ES to converge quickly to a good solution. Further evolved variants make use of a population, by extending the (1+1)-ES to a generic $(\mu+\lambda)$-ES and also include generational replacement, called "comma replacement". Additionally, recombination was introduced into the algorithm, as well as a way to self-adapt the mutation strength during the progress of the search.

4.4 Problem specific operators

The behavior of the mutation operators used in this study is shown in Figure 3. *Uniform1Position* changes the values of one machine in the solution vector to arbitrary values in the whole range. *Swap* mutation exchanges the x,y and rotation coordinates of one machine with that of another, while *RoundedNormalAddition* adds a small value to the positions of each machine and rotates one machine on average in the solution vector.

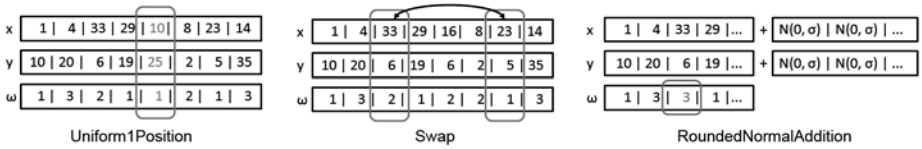

Figure 3: Exemplary graphical description of the three mutation operators

Additionally, two crossover/recombination operators have been used in the study. The first one is a simple *single point crossover* which works similarly to the single point crossover used in the canonical GA: The parents' solution vectors are cut in half at a randomly chosen point and the children composed of the respective halves. The *discrete crossover* performs a cutting principally at every position such that for each machine it is randomly decided which parent genes to use.

5 EXPERIMENTS

We applied the three algorithms on 6 different problem instances of the MPP, three of which are artificial, and three more are adopted from real life data. Situation 1-3 describe facilities with square departments that can be arranged very nicely. The simplest case *4 squares* consists of four squares that are to be placed together such that each of the buffer locations are adjacent to each other and the shape of the whole layout looks like a bigger square. The next situation *8 squares* is slightly more difficult in that the previous situation is doubled. Two blocks of square machines are to be evolved that are not interconnected by flows. Thus the algorithm has to perform a clustering of those machines with flows between them as well as rotate them in an optimal position. The third problem situation *2x8 squares* yet again doubles the previous. The algorithm has to find the optimal layout for 16 squares where there are 4 groups of 4 squares that have stronger flows and two groups of 8 squares with weaker flows. The optimal solution of these instances are not too difficult to find for a human layout planner as the flows are homogeneous within the respective group. The main purpose of these situations is to test an algorithm's performance under controlled conditions. An optimal arrangement of these situations can be seen in Figure 4.

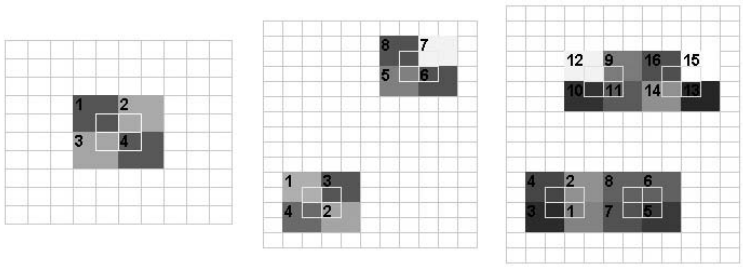

Figure 4: Optimal solutions in the three artificial square problems with 4, 8 and 16 machines. The respective buffer zones are marked by a white border.

Problem situations 4-6 are adopted from real life data and test the algorithm's ability to find layouts under more realistic conditions. In these instances the machines are rectangular as well as square shapes, also including rather extreme shape sizes, such as rectangles with a high width to height ratio. The strength of the flows have been calculated by evaluating a model of the facility under several hypothetical situations. The complexity of the facility is increased by looking at 12, 20 and 30 different machines. The goal of these tests is not to find and prove the effectiveness of a layout for all these situations, but to evaluate the algorithm's ability to find good solutions given the flows of a certain model situation. Each of the three tested algorithms is allowed to run until a certain number of solutions has been evaluated at which time the algorithm is aborted.

The applied variant of the evolution strategy was a (10+200)-ES using a *RoundedNormalAddition* $N(0,\sigma_i)$. The strategy parameter vector $\vec{\sigma}$ was adjusted using two learning parameters τ and τ_0 which were initialized according to the recommendation given in (Beyer and Schwefel 2002). The elements of the initial mutation strength vector were randomly initialized in the interval $[0;\frac{Max(x,y)}{10}[$, with x and y being the dimensions of the floor space. A second tested variant, a (10/2+200)-ES, made use of recombination in the form of discrete recombination for the solution vector and intermediate recombination for the strategy vector $\vec{\sigma}$.

Simulated annealing was configured with a maximum effort of 100 solutions per iteration, an initial temperature of 100, and a multiplicative cooling scheme. The annealing parameter depends on the length of the run and ranged from 0.995 for runs with a maximum of 100,000 evaluations to 0.9997 for runs with a maximum of one million evaluations. The neighborhood was threefold and randomly switched between *Swap*, *Uniform1Position* and *RoundedNormalAddition* $N(0,2)$.

An SGA was tested in four different configurations: Proportional selection and 5% mutation rate (SGA Ia), proportional selection and 10% mutation rate (SGA Ib), tournament selection and 5% mutation rate (SGA IIa), and tournament selection and 10% mutation rate (SGA IIb). The group size of the tournament selector was set to two and 1-elitism was used. The population sizes ranged from 100 to 300 individuals depending on the problem difficulty. The crossover operator for each individual was randomly selected to be either single point or discrete crossover. The mutation operator was the same as the neighborhood operator in SA.

6 RESULTS

The results are given in Table 1, showing the best qualities averaged over 30 runs, as well as the standard deviation. The forth column lists the runtime in seconds of a single run executed on a PC equipped with a Pentium IV 2.8Ghz CPU and 1GB of RAM. The last column contains the maximum number of evaluations before the algorithm is aborted. Figure 7 shows a graphical representation of the best results for the real-world based problem situations. Figure 8 visualizes the strongest flows in one solution found for the problem situation with 20 machines.

Briefly summarizing the results, it turned out that simulated annealing performed best of the three algorithms in 5 out of 6 instances, only on the most simple instance SGA performed better. The genetic algorithm also performed quite well, especially with tournament selection and a slightly higher mutation rate. The results indicate that proportional selection was not working well in combination with a higher mutation rate. The ES did not perform so well on the 6 instances. Performance increased slightly by using recombination, but overall it stayed behind SGA and SA. Simulated annealing also produced results with the lowest standard deviation of all three algorithms and thus is more predictable in providing a good solution. SGA was sometimes able to find better solutions as indicated by the small quality difference to SA and the higher standard deviation, but on average SA was able to converge to a better solution. A two-tailed Mann-Whitney U test has been performed to test the best quality results of SA with the next best performing algorithm (SGA IIb); in all cases except *8 squares* the null hypothesis that the results are equal could be rejected at the 0.05 significance level. Regarding the runtime, ES without recombination was the fastest metaheuristic closely followed by SA. SGA and ES with recombination were approximately equal in runtime.

The first algorithm that we applied to the MPP was the SGA and while it performed quite well, the layouts still showed potential for further improvement in the artificial problem instances. As shown in Figure 5 one group of 8 interconnected machines was arranged almost optimally by our SGA, while the other group still had visible optimization potential. So we applied an evolution strategy to see if the problem could be solved better if mutation was to play a more central role in the optimization process. Tuning the parameters however quickly showed that ES was not able to reach the quality of the SGA. Nevertheless, as the results with tournament selection show, SGA is able to achieve better performance with a higher mutation rate, so the mutation operator does benefit the optimization to a larger degree.

So we applied another mutation-oriented approach and chose simulated annealing. The advantage of SA is that it is among the simpler metaheuristics and therefore quite easy to apply from an implementation standpoint. There is only a single manipulation operator which creates a solution in a neighborhood of another solution. Such a manipulation operator has already been implemented for our SGA in the form of mutation. The results indicate that using the same mutation operators which allowed SGA to converge to good results enabled SA to surpass SGA. Taking a look at the layouts that SA produced revealed that it arranged the machines in a very compact way. However, the question remains why SA was able to perform well, while ES did not. Figure 6 may provide part of the answer to this question in the form of an analysis on the quality progress curves. The quality progress curve of SGA implies that a big part of the search can be attributed to unfocused random search as indicated by the large gap between the best, worst, and average individual and the peaks that sometimes shrink this gap considerably and sometimes increase it to a very high level. The search fluctuates around an elite individual described by the monotone progress in the best solution quality (darkest curve).

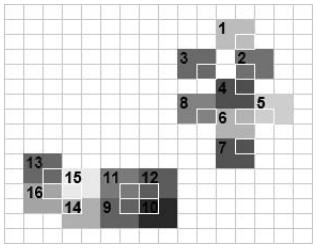

Figure 5: Characteristic final result from our SGA with tournament selection and 10% mutation. One group is arranged close to the optimum while the other group still has visible optimization potential.

The ES on the other hand shows a monotone quality progress in the whole parent population as a result of the plus selection. The children here fluctuate to a lesser degree around the parents and are not allowed to evolve for more than a single generation, except when they are better.

The quality progress of our SA shows that it performs several uphill moves in the beginning, but after about iteration 3000 focuses on improving the current solution as the temperature has dropped such that an uphill move becomes unlikely. So SA balances the diversifying behavior of SGA and the intensifying behavior of ES in the course of its progress.

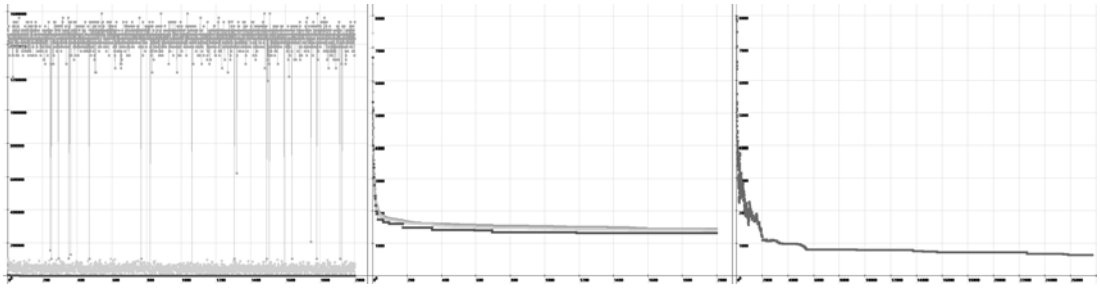

Figure 6: Comparison of the quality progress of SGA (left), ES (middle), and SA (right). The labels on the y-axis range from 200,000 to 1,600,000 in the SGA chart and 2,000 to 8,000 in the ES and SA charts.

Another reason for the performance difference of ES and SA lies in the operators applied. Our ES uses a mutation operator that only moves the machines to a certain degree in a certain direction, while SA also makes use of *swap* and *uniform1position*. Especially *swap* is an operator that increases the neighborhood through exchanges of the position of two machines. Such a move is unlikely to be performed by the addition of a small value to the position, thus including *swap* as an operator of its own is beneficial to the search. But, while the evolution strategy could achieve better results together with *swap*, it showed that this was not the only reason as there was still a significant gap. Tested on the *2x8 squares* problem ES with *swap* achieved an average best quality of 1267.67. So, another factor is the ability of SA to escape from local optima. The evolution strategy with plus replacement is likely to converge to a local optimum very quickly and can only escape this if another better solution is within reach of the neighborhood implicitly created by the mutation operator. We also tried comma selection, which however could achieve only 1859.07 on the *2x8 squares* problem as average best quality over 30 runs. In simulated annealing a move to a worse solution is always accepted with a certain probability, thus the search is less likely to get stuck in a local optimum. In the light of these results, several other metaheuristics might also be of interest, among them tabu search (TS) and variable neighborhood search (VNS). Tabu search features an explicit strategy for escaping local optima, while VNS could exploit to a larger degree the fact that several different mutation/neighborhood operators are already used. Additionally scatter search (SS) could be an interesting approach as it was designed to cover diversification and intensification explicitly. An application of these to the MPP may be the subject of a further study.

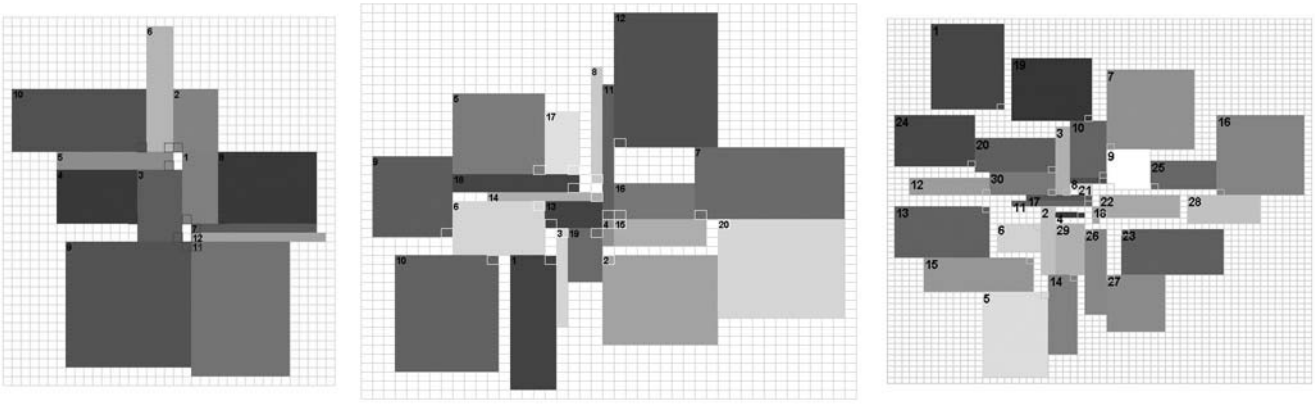

Figure 7: Best solutions obtained from situation 4-6 with 12, 20 and 30 machines

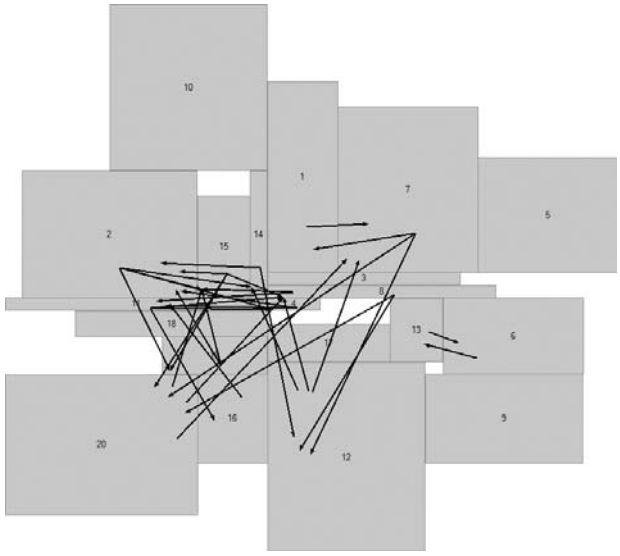

Figure 8: Visualization of only the strongest flows in one solution of the problem situation with 20 machines

7 CONCLUSION AND OUTLOOK

7.1 Conclusion

In this paper we described the machine placement problem and showed how it can be used within an optimization task that involves process simulation to obtain the problem specific data and layout constraints and HeuristicLab to create a floor plan. HeuristicLab as an optimization environment has already been successfully used in various applications, ranging from simulation-based to combinatorial optimization, as well as data mining problems. In this work we focused on the optimization of the MPP and applied three metaheuristics, namely evolution strategies, genetic algorithms, and simulated annealing, to optimize the placement of machines on a production floor with the aim to minimize material transport costs between machine buffer zones.

As demonstrated in Sections 5 and 6, a considerable amount of testing and refining was necessary to find an approach and parameterization that works well on different MPP instances. We found that simulated annealing out-performed the standard genetic algorithm and evolution strategies with and without recombination in 5 out of 6 instances and attributed it

Table 1: Comparison of best quality results obtained from several optimization strategies on a number of problem situations

Problem	Algorithm	Average Best Quality	Runtime [s]	Evaluations
4 squares	(10+200)-ES	23.40±3.53	20.67	100,000
	(10/2+200)-ES	22.07±3.46	32.84	100,000
	SA	21.67±2.17	30.76	100,000
	SGA Ia	19.13±4.63	31.61	100,000
	SGA Ib	21.20±4.32	31.72	100,000
	SGA IIa	20.20±4.11	32.96	100,000
	SGA IIb	**18.47**±3.43	33.33	100,000
8 squares	(10+200)-ES	73.33±9.31	43.69	200,000
	(10/2+200)-ES	72.60±8.63	68.64	200,000
	SA	**43.27**±4.11	59.42	200,000
	SGA Ia	46.27±8.75	65.54	200,000
	SGA Ib	66.20±11.98	65.07	200,000
	SGA IIa	45.27±6.02	68.54	200,000
	SGA IIb	43.80±5.39	68.10	200,000
2x8 squares	(10+200)-ES	1658.00±289.03	105.82	400,000
	(10/2+200)-ES	1411.20±72.46	155.40	400,000
	SA	**686.40**±29.57	131.09	400,000
	SGA Ia	836.27±113.95	146.82	400,000
	SGA Ib	1165.73±146.55	146.98	400,000
	SGA IIa	751.60±60.00	152.29	400,000
	SGA IIb	731.20±78.93	153.81	400,000
situation 4	(10+200)-ES	6746.73±964.76	78.59	300,000
	(10/2+200)-ES	6015.14±722.12	115.12	300,000
	SA	**3018.41**±238.84	95.69	300,000
	SGA Ia	4002.03±419.23	109.31	300,000
	SGA Ib	5045.95±559.68	109.39	300,000
	SGA IIa	4346.77±633.63	113.85	300,000
	SGA IIb	3848.71±476.21	114.60	300,000
situation 5	(10+200)-ES	86702.57±73064.37	192.75	600,000
	(10/2+200)-ES	46551.86±14029.37	274.94	600,000
	SA	**12481.25**±625.30	224.00	600,000
	SGA Ia	21346.35±1807.69	256.76	600,000
	SGA Ib	31747.84±2828.18	259.56	600,000
	SGA IIa	17234.72±1351.57	260.43	600,000
	SGA IIb	15728.18±1126.74	266.02	600,000
situation 6	(10+200)-ES	152134.05±98886.87	465.48	1,000,000
	(10/2+200)-ES	159869.78±134391.76	616.69	1,000,000
	SA	**39314.40**±1336.66	490.12	1,000,000
	SGA Ia	95756.29±4927.72	569.51	1,000,000
	SGA Ib	121004.05±6252.40	576.25	1,000,000
	SGA IIa	50261.10±2782.83	572.30	1,000,000
	SGA IIb	46735.43±2040.03	574.82	1,000,000

to the algorithm's ability to escape local optima. Moreover, we identified additional metaheuristics that might perform well on the MPP such as tabu search and variable neighborhood search.

7.2 Outlook

Our plans for future research activities revolve around two objectives. First and foremost we wish to further extend and refine the MPP together with our partners and associated partners. Possible extensions are the addition of separate buffer zones for incoming and outgoing material, the inclusion of floor space utilization into the cost function, the introduction of further constraints that are necessary for the creation of a feasible layout plan, and more. However, we do not plan to evolve the MPP into a feature-rich, highly specialized problem description, but aim to find relevant abstractions of real-world situations. In our view, problem descriptions like the MPP are interfaces between an optimization environment and applications on a potential real-world problem, as in this case layout problems in production environments. Creating such interfaces enables the interaction of the problem domain, which can be modeled in simulation environments, data tables, state charts, or process descriptions, and the optimization domain, which can be modeled via optimization algorithms, expert systems, and analysis tools. To create such interfaces and to gain knowledge regarding their implementation is an important step to achieve our second objective:

A growing number of problems is solved by computer aided optimization every year, and thus attracts an increasing number of people working in this area. Simulation can be regarded as a popular way of modeling several such problems and has thus increased the number of potential applications even more. Knowledge about the application of optimization algorithms is generated everywhere in the world, but often enough used in only one or two projects and typically not processed

in such a way that it can be readily shared with other practitioners. Our long term goal is to store and reuse the results that we derive from our applications and projects in an automated way. HeuristicLab is an environment for modeling optimization algorithms instead of programming them; as such we will be looking into ways on how to compare these algorithm models and how to automatically derive performance measures based on the structure and parametrization of these models. The user of this environment will not only benefit from the availability of ready-to-use optimization methodologies, but also from the possibility to build on knowledge that is and has already been found.

ACKNOWLEDGMENTS

The work described in this paper was carried out within the Josef Ressel Centre for Heuristic Optimization and supported by the Austrian Research Promotion Agency (FFG).

REFERENCES

Affenzeller, M., G. Kronberger, S. Winkler, M. Ionescu, and S. Wagner. 2007. Heuristic optimization methods for the tuning of input parameters of simulation models. In *Proceedings of I3M 2007*, 278–283: DIPTEM University of Genova.

Affenzeller, M., S. Winkler, S. Wagner, and A. Beham. 2009. *Genetic algorithms and genetic programming - Modern concepts and practical applications*. Numerical Insights. CRC Press.

Beham, A., M. Affenzeller, S. Wagner, and G. Kronberger. 2008. Simulation optimization with HeuristicLab. In *Proceedings of the 20th European Modeling and Simulation Symposium (EMSS2008)*, ed. A. Bruzzone, F. Longo, M. Piera, R. Aguilar, and C. Frydman, 75–80: DIPTEM University of Genova.

Beham, A., M. Kofler, M. Affenzeller, and S. Wagner. 2009. Evolutionary selection in simulation-based optimization. In *12th International Conference on Computer Aided Systems Theory EUROCAST 2009*, ed. A. Quesada-Arencibia, J. C. Rodrguez, R. Moreno-Diaz jr., and R. Moreno-Diaz, 262–263: IUCTC Universidad de Las Palmas de Gran Canaria.

Beham, A., S. Winkler, S. Wagner, and M. Affenzeller. 2008. A genetic programming approach to solve scheduling problems with parallel simulation. In *Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS08)*: IEEE.

Beyer, H.-G., and H.-P. Schwefel. 2002, March. Evolution strategies - A comprehensive introduction. *Natural Computing* 1 (1): 3–52.

Burkard, R., E. Cela, S. Karisch, and F. Rendl. 2009, January. QAPLIB - A quadratic assignment problem library. Available via <http://www.seas.upenn.edu/qaplib/> [accessed April 9, 2009].

Burkard, R. E., S. E. Karisch, and F. Rendl. 1997, June. QAPLIB A quadratic assignment problem library. *Journal of Global Optimization* 10 (4): 391–403.

Can, B., C. Heavey, and A. Beham. 2008a. A comparative study of genetic algorithm components in simulation-based optimisation. In *Proceedings of the 2008 Winter Simulation Conference*, ed. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 1829–1837: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers,Inc.

Can, B., C. Heavey, and A. Beham. 2008b. Simulation-based evolutionary optimisation of buffer allocation problem. In *Proceedings of the Operational Research Society 4th Simulation Workshop (SW08)*.

Carson, J. S. 1997, December. AutoStat: Output statistical analysis for AutoMod users. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, 649–656. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Doerner, K. F., M. Gendreau, P. Greistorfer, W. Gutjahr, R. F. Hartl, and M. Reimann. (Eds.) 2007. *Metaheuristics: Progress in complex systems optimization*. Operations Research/Computer Science Interfaces Series. Springer.

Drezner, Z., P. M. Hahn, and E. D. Taillard. 2005. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult to solve for meta-heuristic methods. *Annals of Operations Research* 139:65–94.

Dumitrescu, D., B. Lazzerini, L. C. Jain, and A. Dumitrescu. 2000. *Evolutionary computation*. The CRC Press International Series on Computational Intelligence. CRC Press.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

Hahn, P. M., and J. Krarup. 2001. A hospital facility layout problem finally solved. *Journal of Intelligent Manufacturing* 12:487–496.

Holland, J. H. 1975. *Adaption in natural and artificial systems*. University of Michigan Press.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220:671–680.

Kofler, M., A. Beham, M. Affenzeller, and S. Wagner. 2008. Optimization of medical ultrasound transducers with simulation and genetic algorithms. In *Proceedings of the 20th European Modeling and Simulation Symposium (EMSS2008)*, ed. A. Bruzzone, F. Longo, M. A. Piera, R. M. Aguilar, and C. Frydman, 100–105: DIPTEM University of Genova.

Koopmans, T. C., and M. Beckmann. 1957, January. Assignment problems and the location of economic activities. *Econometrica, Journal of the Econometric Society* 25 (1): 53–76.

Lodi, A., S. Martello, and M. Monaci. 2002. Two-dimensional packing problems: A survey. *European Journal of Operational Research* 141 (2): 241–252.

McKendall, Jr., A. R., J. Shang, and S. Kuppusamy. 2006. Simulated annealing heuristics for the dynamic facility layout problem. *Computers & Operations Research* 33 (8): 2431–2444.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21:1087–1092.

Rawles, I. 1998. The WITNESS toolbox - A tutorial. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D. Medeiros, E. Watson, J. Carson, and M. Manivannan, 223–226. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rechenberg, I. 1973. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog.

Sadowski, D., and V. Bapat. 1999. The Arena product family: Enterprise modeling solutions. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P. Farrington, H. Nembhard, D. Sturrock, and G. Evans, 159–166. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sahni, S., and T. Gonzalez. 1976. P-complete approximation problems. *Journal of ACM (JACM)* 23 (3): 555–565.

Wagner, S. 2009. *Heuristic optimization software systems - Modeling of heuristic optimization algorithms in the HeuristicLab software environment*. Ph. D. thesis, Johannes Kepler University, Linz, Austria.

Wagner, S., G. Kronberger, A. Beham, S. Winkler, and M. Affenzeller. 2008. Modeling of heuristic optimization algorithms. In *Proceedings of the 20th European Modeling and Simulation Symposium*, ed. A. Bruzzone, F. Longo, M. A. Piera, R. M. Aguilar, and C. Frydman, 106–111: DIPTEM University of Genova.

Wagner, S., S. Winkler, R. Braune, G. Kronberger, A. Beham, and M. Affenzeller. 2007. Benefits of plugin-based heuristic optimization software systems. In *Computer Aided Systems Theory - EUROCAST 2007*, ed. R. Moreno-Diaz, F. Pichler, and A. Quesada-Arencibia, Volume 4739 of *Lecture Notes in Computer Science*, 747–754: Springer.

Wolpert, D. H., and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1 (1): 67–82.

Yeniay, O. 2005. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications* 10 (1): 45–56.

AUTHOR BIOGRAPHIES

ANDREAS BEHAM received the MSc in computer science in 2007 from Johannes Kepler University (JKU) Linz, Austria. His research interests include heuristic optimization methods and simulation-based as well as combinatorial optimization. Currently he is a PhD student at the JKU and research associate at the Research Center Hagenberg of the Upper Austria University of Applied Sciences. He can be contacted via <abeham@heuristiclab.com>.

MONIKA KOFLER is a research associate at the Research Center Hagenberg and currently pursues her PhD in heuristic optimization and production planning. In 2006 she graduated with a diploma's degree in Medical Software Engineering from the Upper Austria University of Applied Sciences. Monika Kofler's contact email address is <mkofler@heuristiclab.com>.

STEFAN WAGNER currently holds the position of an Associate Professor of Software Engineering at the Upper Austria University of Applied Sciences. In 2004 he received the MSc in computer science, and in 2009 the PhD in engineering sciences, both from Johannes Kepler University Linz, Austria. Stefan Wagner is the chief architect of HeuristicLab and conducts active research on evolutionary computation, heuristic optimization, theory and application of genetic algorithms, machine learning and software engineering. For further information contact him at <swagner@heuristiclab.com>.

MICHAEL AFFENZELLER has published several papers and journal articles dealing with theoretical aspects of evolutionary computation and genetic algorithms. His professional activities include the organization of several workshops in the field of heuristic optimization and reviewing activities for relevant journals. In 2001 he received the PhD in engineering sciences and in 2005 the habilitation in applied systems science, both from the JKU Linz, Austria. Since 2005 he is professor at

the Upper Austria University of Applied Sciences, Campus Hagenberg, and head of the Josef Ressel Center Heureka! at Hagenberg. Michael Affenzeller can be contacted via <maffenze@heuristiclab.com>.

The web pages of all members of the *Heuristic and Evolutionary Algorithms Laboratory* as well as further information about HeuristicLab and related scientific work can be found at: <http://www.heuristiclab.com>