Adaptive Mutation Rate at Genetic Algorithms with Multi-Chromosome Representation in Multi-department Hospital Process Optimization

Matthias Kühn
Technical University Ilmenau
Ehrenbergstraße 29
98693 Ilmenau, Germany

Thomas Severin
Technical University Ilmenau
Ehrenbergstraße 29
98693 Ilmenau, Germany

Joachim Lippold
Technical University Ilmenau
Ehrenbergstraße 29
98693 Ilmenau, Germany

Horst Salzwedel
MLDesign Techn. Inc.
2230 Saint Francis Drive
CA 94303 Palo Alto, USA

Volker Nissen
Technical University Ilmenau
Helmholzplatz 3
98693 Ilmenau, Germany

ABSTRACT
The performance of Metaheuristics in general and Evolutionary Algorithms (EA) in particular depends on good settings of algorithm parameter values, such as population size, mutation rate or crossover probability. To increase performance, researchers still try to find optimal settings. At present, researchers are adapting the parameter settings during an evolutionary run (parameter control). Thus, no hand tuning is needed upfront of an evolutionary run. In this paper we analyze algorithm performance when using adaptable algorithm parameters on Genetic Algorithms (GA) with multi-chromosome representation. Most of the research in the field of EA has been done on a theoretical basis. Often the proposed solutions do not deliver what they promise, when applying them to complex problems of real-world. Thus, experimental studies on complex problems of real-world are needed to ascertain performance improvement of adaptive parameter control. This paper is an experimental study on such a complex optimization problem of real-world (dynamically coupled System of Systems). In our approach of parameter control new individuals are generated by adapting the mutation rate. Therefore, we calculate a dedicated mutation rate for each chromosome of the individual. This happens in relation to the fitness of each chromosome. We analyzed and have statistically proven the outperformance of our approach upfront with the De Jong's (Sphere) and the Schwefel's test function. In this paper, we are now applying our approach to a real world based complex optimization problem (nonstationary, dynamic, noisy), to prove the outperformance of our approach. Therefore, we made a performance comparison with non-adaptive GA, which demonstrates the superiority of the adaptive approach. More specifically, we use a stochastic simulation model of university hospital processes. Inpatient admission, outpatient admission and op-theater planning of elective patients must be optimized simultaneously, while emergencies occur. Every hospital area has its own objectives and constraints (dedicated systems). The number of patients and utilization of resources must be maximized in every hospital area, while waiting times, lead times and schedule variances must be minimized. In that, a system of systems can be seen. It is shown how our approach can be used to optimize such dynamically coupled system of systems (SoS) in an efficient way.

Keywords
Genetic algorithms, hospital, inpatient admission, multi-chromosome, mutation rate, op-theater planning. optimization, outpatient admission, parameter control, self-adapting, computer simulation, real world problem, system of systems optimization.

1. INTRODUCTION
Since Genetic Algorithms (GAs) were introduced by Holland [1], GAs were increasingly used to solve optimization problems when exact, analytic methods are not available or cannot be applied. Since the early beginning researchers try to find optimal parameter settings for GA control parameters like, population size, crossover probability or mutation rate. By tuning these control parameters better solutions can be found in less time. Several researchers like De Jong [3] or Schaffer et al. [4] focused on finding optimal algorithm parameter settings. But the results are specific to the optimization problems of their test cases and cannot be generalized [5, p. 124 f.]. Finding the optimal values for specific problems or in general is a long-standing challenge in the field of Evolutionary Algorithms (EA). In the past parameter settings were tuned upfront the optimization run (evolutionary run). But tuning of GA control parameter upfront is very time consuming. At present researchers ar focusing on parameter control, which is adapting the control parameters during evolutionary run, to overcome the problem of upfront time-consuming hand tuning or using given parameter settings of other test cases. Parameter control is "still in its infancy, requiring fundamental research" [5, p. 146], e.g. towards good control strategies. Thus, looking at GAs, it is necessary to focus on selected parameters to find effective ways of parameter control. It is difficult to ascertain witch parameter control strategy would improve performance [5, p. 63 f.]. EAs and also GAs are stochastic, non-linear algorithms. Formal proof is extremely difficult [2, p. 6]. A deeper understanding of how changes in GA parameters affect GA performance can be obtained by experimental studies, e.g. setting one parameter adaptive and keeping the other values fixed throughout the run. Most of the research in the field of EA has been done on a theoretical basis. Often the proposed solutions do not deliver what they promise, when applying them to complex problems of real-world. Thus, experimental studies on complex problems of real-world are needed to ascertain performance improvement of adaptive parameter control. This paper is an experimental study on such a

complex real-world optimization problem (dynamically coupled system of systems). At a glance, this paper is focusing on five issues:

- adaptive parameter control
- GA with multi-chromosomal representation,
- real-word optimization problem (nonstationary, dynamic, noisy)
- multi-objectives (multi-system optimization: dynamically coupled system of systems)
- obtaining performance improvements

In the following section II related work is given. In section III we describe a real-world case. Section IV points out the experimental design and specifies the test scenarios. In section V we present the results. Section VI summarizes the work and gives an outlook on future work that needs to be done.

2. RELATED WORK

Early research by De Jong [3] or Schaffer et al. [4] focused on finding optimal algorithm parameter settings. These results are specific to the optimization problems of their test cases and cannot be generalized [5, p. 124 f.]. The parameter must be specified at the beginning and remain static during the evolutionary run. This is the commonly practiced approach called parameter tuning [12, p. 20]. It means to figure out the best values in preliminary runs. It is very time consuming and computationally expensive.

The Evolutions Strategy (ES) by Rechenberg and Schwefel led to the development of adapting control parameter [9]-[11]. This forms an alternative, called parameter control, where parameter values are changed dynamically during runs [12, p. 20]. Eiben, Hinterding & Michalewicz [6, p. 131] distinguish three types of parameter control (see Figure 1):

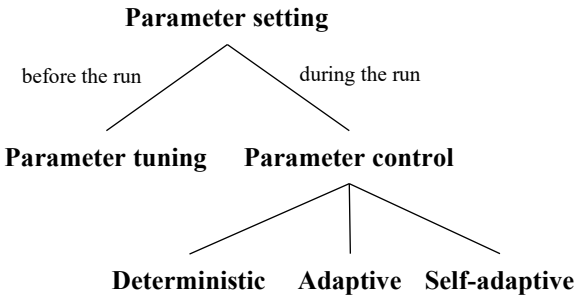

Figure 1. Global taxonomy of parameter setting in EA's [6, p. 129]

Deterministic, as a blind deterministic rule, e.g. triggered by the process of time (number of generations). Adaptive parameter control, which incorporate feedback from the search process, like Rechenberg's "1/5" success rule [10]. And self-adaptive, when using a meta-EA or by using EA that tunes itself to a given problem. It is typically done via a mechanism in which the algorithm parameter values are encoded as a control gene on individual genomes [2, p. 7].

Aldeida & Moser [40], De Jong [2], Eiben & Smit [36], Karafotias et al. [37] and Meyer-Nieberg & Beyer [35] are giving an overview of parameter control for EA. Today, parameter control is a standard component of many ES algorithms [2, p. 13]. Except of ES, for mutation step size adaption, none of the EAs are using adaption routinely in every day practice [2, p. 6]. Adapting control parameter during EA run, e.g. mutation rate, is also not common for Genetic Algorithms [2, p. 13]; [35, p. 48]. However it has been empirically and theoretically demonstrated that different values of parameters at different stages of the evolutionary process increase algorithm performance [12, p. 21, 41]; [5, p. 58, 131], for example in [13]-[20], [38], [41], [43]. Fogarty [13] uses a deterministic rule and has shown that reducing mutation rate exponentially over time increases the performance of the GA. Hesser & Männer [14] obtained the result of Fogarty that mutation rate should be decreased during convergence. Fernandes et al. [38] introduced a method for GAs mutation rate control, based on the Sandpile Model. The Sandpile is a complex system operating at a critical state between chaos and order. The mutation intensity changes along the search process and also depends on the convergence stage of the algorithm. The approach of sandpile mutation must work on evaluated individuals that would require two cycles of evaluations per generation. To avoid this, Fernandes et al. use the fitness values of the parents of an individual to derive an expected normalized fitness. They say that Sandpile Mutation appears to be well suited for function optimization in dynamic environments. Grefenstette [15] uses an metalevel GA to find optimal values for six algorithm parameters, like crossover and mutation rate. An additional GA was used to identify optimal algorithm parameter values of another (subjected) GA. The idea of the meta-GA was revived by Yuan & Gallagher [43] on a hybrid approach with racing scheme. Bäck [16], [17] also uses self-adaption and handled mutation rate as temporal and individually differing parameter, which is incorporated into the genetic representation of the individuals. He confirmed that mechanism of self-adaptive mutation rate is advantageous for GAs performance. Bäck [16] sees this as a strong argument for general introduction of adaptive mutation rates to GAs. Smith [18] improved the scheme of Bäck, by using a fixed learning rate instead of also variate the rate of variation. Serpell & Smith [41] uses self-adaption to adapt the choice of mutation operator as well as the mutation rate for the chosen operator during runtime. Serpell & Smith showed that all the tested self-adaptive GAs provided comparable or better results to the best choice of non-adaptive GAs. But, they have seen an overhead at self-adaption in the time required to find the optimal mutation rate (costs of self-adaption). Further on, they imply that self-adaptation of the mutation rate takes precedence over the self-adaptation of the mutation operator.

Just a little research is done on adapting algorithm parameters for GAs with multi-chromosome representations. It is a fact, that complex optimization problems of real-world usually have more than one objective and can be seen as system of systems, with each of the containing systems having its own characteristics. Therefore mu1ti-chromosome representations seem to be very suitable. They have been used in GAs to encode different aspects of the representation of a problem (solution) onto separate chromosomes. Thus, it is possible to decompose a problem (solution) into several simpler parts so that each part can be represented onto a separate chromosome. Each chromosome can use a different representation and its own set of reproduction operators. From our point of view this is needed to efficiently represent and optimize system of systems and also to find effective adaptive control strategies for algorithm parameters.

Hinterding [19] used two chromosomes to investigate the effect of self-adaptive mutation rate. He used one Chromosome to represent the problem (solution) and the other one to represent the self-adaptive strategy. Hinterding showed that self-adaptation leads to better results, except for the easier problems run for shorter numbers of evaluations. He points out that this could be attributed to the fact that the self-

adaptive parameters were initialized with uniform random values, and the GA needs some time to evolve appropriate values for them. Hence, in that he has also seen the cost for using self-adaptation. But Hinterding says that the costs are out-weighed by the better results for the harder problems and remove the need to hand-tune the parameters that are adapted.

Kühn et al. [20] introduced an approach of adaptive parameter control of mutation rate for GAs with multi-chromosome representation. The test suite was given by De Jong's (Sphere) and Schwefel's function. The optimization problem was divided into two parts and represented by two chromosomes. For example, at Schwefel's function (SF) [10]:

$$f_S(\vec{x}) = \sum_{i=1}^{n} -x_i * \sin(\sqrt{|x_i|} \qquad (1)$$
$$-500 \leq x_i \leq 500$$
$$n \in N;\ \overrightarrow{x^*} = (x_1^*, x_2^*, x_3^*, \dots, x_n^*)$$

The parts are (2, 3):

$$f_{S_1}(x_1) = -x_1 * \sin(\sqrt{|x_1|}), \qquad (2)$$

$$f_{S_2}(x_2) = -x_2 * \sin(\sqrt{|x_2|}). \qquad (3)$$

There is one chromosome representing x_1 and the other one x_2. When generating new individuals by mutation, for each chromosome of the selected individual a dedicated mutation rate is calculated. Therefore, an additional fitness value for each chromosome (chromosome fitness) is calculated. This value is used to adapt the mutation rate to the fitness of each chromosome. Better fitness of a chromosome leads to lower mutation rate. Thus, the mutation rate can vary between the chromosomes of an individual and in every generation (over time). To calculate the specific mutation rate of a chromosome in relation to the chromosome fitness either a linear or an exponential function was used. While fitness is not that good, like at the beginning of the GA run, the resulting mutation rate is very high with 0.5 in maximum. The reason for this is seen in a better exploration of the search space at the beginning of a search process to efficient locate regions with good fitness values. Therefore, large mutation rates are needed [12, p. 21] [5, p. 131]. Later, when only fine tuning has to be done smaller mutation rates are required. While the fitness level is getting better over time, the mutation rate decreases to 0.001 in minimum. The test functions mentioned above were used to analyze the GA performance. GA runs with non-adaptive mutation rates of 0.001, 0.01, 0.05, 0.1, 0.2 and 0.5 were compared to the approach of adaptive mutation rate. All other algorithm parameters remained the same during the run. Population size was set to 20 individuals. Thus, a quite small population was chosen. The reason is that on complex optimization problems long run times may be expected when having a large population size. Less individuals might lead to less run time and computation expenses. Every individual consists of two binary coded chromosomes each representing half of the search space (resp. problem solution). Selection is done by roulette wheel. A normalization for the fitness values was done, according to the parent population. Single point crossover was used at every chromosome. Crossover probability was set to 0.60. At every variant of mutation rate 100.000 runs were performed using De Jong's and Schwefel's function. The GA run ended when a fitness level of 0.995 was reached. The number of generations it took to reach the defined fitness level was measured and statistically analyzed. The mean value for each variant was calculated. Kühn et al. [20] have shown for GA with multi-chromosome representations that adaptive mutation rate based on chromosome fitness leads to significant better GA Performance, compared to non-adaptive variants. It was shown that a linear function to calculate mutation rates performed much better than exponential. For linear mutation function the duration to reach a defined fitness level was decreased by at least 38 % compared to the best non-adaptive variant. In only one variant of Sphere function, the results are worse than in best non-adaptive run. The reason can be seen in the easy test function. The high mutation rate at the beginning leads to better exploration of the search space, what might not be needed at that easy (static) problem. That also corresponds to De Jong [2, p. 16], when he is saying non-adaptive algorithm parameters "that have been pre-tuned for particular classes of problems will continue to outperform". Kühn et al. say, that their approach needs to be wider tested, e.g. on complex optimization problems of real world. This will be done in the following.

3. REAL WORLD CASE AND REPRESENTATION

A complex problem of real-world can be seen as an optimization problem that can be found in real-world, with dynamic behavior, noisy and nonstationary conditions. Such a problem cannot be solved analytically. In complex real-world optimization problems, usually more than one objective needs to be optimized. Optimization problems consisting of several systems that are dynamically coupled and interacting with each other (system of systems) are increasing in number. In general, systems can be differentiated e.g. on different objectives, different resources, different constraints or even different languages that are spoken. Some researcher is merging these systems on a high level of abstraction to make optimization easier. That might end in solutions that do not have expected results, when applying them to the real-world. That is why there is the need and acuteness to figure out efficient ways to optimize complex optimization problems of real-world, in particular for dynamic, coupled system of systems. An example for that kind of complex optimization problems can be seen in the inpatient admission, outpatient admission and op-theater planning of elective cardiologic patients of a German university hospital, which must be optimized together (scheduling and sequencing) while emergencies occur. The number of patients treated should be maximized, patient waiting times and idle times of hospital resources should be minimized. Considered hospital entities are: outpatient department, two cardiological wards, three heart catheter laboratories (HCLs) as op-theaters, an electro-physiological laboratory (op-theater), an electrocardiographic unit (ECG) and three echocardiographic units (ECHO). Each of these units can be seen as a system. They are dynamically coupled and each of them has their own objectives and constraints of planning scheduling and sequencing. Moreover, diverse uncertainties (noise) need to be considered, like patient's health conditions, duration of examinations, lateness of patients, decisions made by physicians and emergencies that occur. On planning and optimization, downstream resources and dynamic of processes must be considered. Downstream resources are resources that patients went through after admission. None of the three university hospitals analyzed have noise conditions or downstream resources considered at admission planning. Even at op-theater planning no emergencies were considered.

By now, no research is seen, considering this together [46]-[50]. Also, no analytic way is seen as suitable to calculate an

optimal solution. The reason can be the fact, that we think it is not possible to put all the objects to be planned, all constraints, all uncertainty and the dynamic behavior in a mathematic function. In case it is possible somehow, we are not able to say that the optimization function is continuously differentiable twice. Nor we do see a way to calculate every single possible solution of the search space. Knowing that there is a huge number of possible combinations and knowing that by the noise, later we will get different results every time, when testing a solution.

To analyze or optimize highly dynamic processes as can be found within hospitals, a dynamic executable model is necessary. Further on a heuristic is needed to optimize it. We have tested a GA in Kühn at al. [53] and think it is suitable to the given optimization problem. Following this, we use the stochastic, executable simulation model developed by Kühn & Lippold [21] to optimize the hospital processes. The simulation model was built in MLDesigner as a discrete event model. Stochastic effects within the model are, e.g.:

- unplanned patients (walk-in),
- emergency patients from emergency department,
- occurring of an emergency while a treatment is running or within bedtime on a ward,
- unpunctuality of patients,
- upfront unknown treatment plan,
- variation in duration of treatments,
- fault of medical devices.

For purpose of modelling a database containing 2 years of collected real-world data (SAP IS-H*med extraction, empirical data collections) and process descriptions of all the related hospital processes were used. Hospital processes were automatically transformed and imported into the simulation model, following the approach of Kühn at al. [44]. The database of real data has been divided into two parts following Page [45, p.149]. The first part was only used for modelling and parameterizing the simulation model. The second part of real-world data was used to validate the simulation model. As a result, a validated, dynamic simulation model is developed, which includes all the mentioned constraints, uncertainties and given capacities of considered hospital resources. Now an optimization of admission planning, patient scheduling and sequencing can be done by simulation possible solutions. On evolution run, solutions will be created, tested and evolve, fitting to the environment.

For the simulation model Severin [39] implemented a GA, with multi-chromosome representation. It appeared quite difficult for us to merge all planning parameters on one chromosome. Dividing the problem representation by units (systems) and planning focus on dedicated chromosomes makes it much easier for us to represent the different optimization objectives in an appropriate way. Further on, the chromosome structure differs on the specific planning focus.

In detail, the chosen optimization problem is represented by 15 chromosomes. 7 of these chromosomes are used for patient admission planning. For each hospital area that needs to be considered (e.g. outpatient department, wards, ECG, ECHO, HCLs), a dedicated chromosome (CZ01-CZ07) is used to plan the number of patients and day of treatment/admission (day of week). Table 1 shows the structure of these seven chromosomes. All of them are dynamic in length. So, the number of patients planned is represented by the length of the chromosome, which changes during the EA run.

Table 1: Structure of chromosomes to plan number of patients and the day of admission (CZ01-CZ07)

Patient_ID	01	02	03	04	05	…	21	22	23	24	..
Day of Week	1	1	3	1	1	…	3	1	4	4	..

Two of the mentioned 15 chromosomes are used to plan the appointment rules and start time of admission (opening time) for the outpatient department (CT01) and the considered wards (CT02). At these two chromosomes, optimization parameters are n_1: number of patients at first appointment, n_i: number of patients at each appointment; a_i: time between two appointments, following Cayirli & Veral [46], and time of the first appointment (begin opening time). Table 2 shows the structure of these two chromosomes.

Table 2: Structure of chromosomes for inpatient and outpatient admission (CT01+CT02)

Gen	n_1	n_i	a_i	Mo_begin	Tu_begin	We_begin	Th_begin	Fr_begin
value	1	1	15	450	510	510	510	570

Planning purpose for ECGs, ECHOs and HCLs (op-theaters) is only the number of patients and day of appointment. The appointment times are assigned by an implemented appointment calendar within the hospital model, that is initiated with e.g. frequently planned maintenances and all the other times, when appointments cannot be made, to ensure that appointments do not overlap and will be made at valid times. In general appointments from calendar are given by first-come, first-appointment rule. This can vary in case a patient already has other appointments. Thereby a period of time is given (before and afterwards) within that no other appointment is made.

The remaining six chromosomes are used to plan reservations upfront the admission and to plan sequencing and scheduling of patients in ECGs, ECHOs and OP-Theaters units. The following table 3 shows the structure of these six chromosomes. The intention is to make reservation to defined types of patients upfront the admission and to minimize patient waiting time at wards after admission. Because, a lot of times patients waiting on ward for treatment, while resources needed for treatment (like ECG, ECHO or HCL) were not available that soon after admission.

Table 3: Structure of chromosomes for reservation planning in ECG, ECHOs and HCLs

slot number	01	02	03	04	…	18	19	…
value	0	1	1	3	…	0	2	…

The length of these chromosomes is defined by the number of

available time slots within the mentioned calendar in the simulation model. Each slot can be set to a value of 0 up to 3. Only the value is manipulated by the GA. A value of 0 is representing no reservation. Thus, all patients can make an appointment at this time slot. Values from 1 to 3 are reservations made upfront and dedicated to a defined group of patients, like patients on ward or outpatients.

By differentiating in chromosomes, it was possible to easily implement the genetic operators independent and individually, in regard to the need of the different chromosome structure. That was much easier than having one operator to handle all cases and constraints at once, in a single chromosome representation. A huge difference between the chromosomes was to handle mutation and crossover operators on chromosomes representing the number of patients, that are varying in length.

4. ADAPTING GA FOR PARAMETER CONTROL

Looking at the algorithm performance, relating to the chosen complex real-world problem, parameter tuning does not appear practical. Testing one solution only takes 17 seconds. But testing 200 generations with a population of 30 individuals and testing each solution 10 times (because of noise), it takes 11,8 days! for only one evolutionary run. Thus, preliminary runs are too much time consuming. That is why an adaptive control strategy is used in the following.

Eiben [5, p. 145] sees the research community to converge on the idea that: "successful parameter control must take into account two types of information regarding the evolutionary search: data about fitness and population diversity." Mutation rate can be seen as a control mechanism to preserve diversity of the population. Moreover, the mutation rate seems to be a very sensitive parameter with high impact on efficiency of GA [35, p. 59]; [4, p. 59]; [10, p. 7]; [7, p. 228]. That is why this paper is focusing on the mutation rate in the following. When adapting the GA, the approach of Kühn et al. [20] is used. Both the mutation rate (diversity) and data about fitness are considered in this approach (see details in the former section 2).

As already explained, this approach includes at every generation an additional fitness rating for each chromosome. When optimizing system of systems, it appears quite difficult to merge all optimization objectives into one objective function. By having an additional chromosome fitness, chromosome specific objectives can be applied to the corresponding chromosomes and later on main objectives to the individual. Thus, a hierarchy of objectives is given. First a dedicated fitness function for each of the 15 chromosomes needs to be defined, based on the constraints and objectives of the dedicated planning purpose. Here are some examples, showing the way it was to be handled. All of the mentioned key figures are calculated excluding settling time.

CZ01: Planning purpose is the number of patients and the admission day (day of week) for the outpatient department.

- The number of outpatients admitted a day in the outpatient department (mean per week) has to be maximized.

$$tFP1 = min\left(1.0; \frac{OutPat}{Weeks * Patients_{max}}\right) \quad (4)$$

- Lead times (Length of stay) of outpatients admitted in the outpatient department (mean per patient) has to be maximized.

$$tFP2 = \max\left(0.0; 1 - \frac{OutLoS}{OutPat * OutLoS_{max}}\right) \quad (5)$$

- Number of outpatients admitted in the outpatient department, that have not finished by end of day (mean per Week) has to be minimized.

$$tFP3 = \max\left(0.0; 1 - 2 * OutPat_{inc} * \frac{OutLoS_{mean}}{Weeks * Patients_{max}}\right) \quad (6)$$

OutPat : outpatient admitted in the outpatient department
Weeks: number of weeks simulated
$Patients_{max}$: Maximum number of Patients per week possible
OutLoS: patient length of stay in the outpatient department
$OutLoS_{max}$: Maximum OutLoS possible
$OutPat_{inc}$: number of patients in the outpatient department that have not left at the end of day

The three key figures (tFP1-tFP3) mentioned above are equal weighted in the chromosome fitness function.

$$F_{CZ01} = \frac{tFP1 + tFP2 + tFP3}{3} \quad (7)$$

CZ02: Planning purpose is the number of patients and day of admission (day of week) for inpatients on ward 1.

- The number of inpatients admitted at ward 1 a day (mean per week) has to be maximized.
- Waiting time of inpatients admitted for a bed on ward (mean per patient) has to be minimized.
- Number of patients admitted, that do not have a bed on ward at the end of the day (mean per week) have to be minimized.

The three key figures mentioned are equal weighted in the chromosome fitness function.

CT01: Planning purpose is the appointment rule for outpatient department and period of time where appointments are made for each day of the week.

- Waiting time of outpatient admitted in the outpatient department till first contact (after admission) (mean per patient) has to be minimized.
- Lead time of outpatient admitted in the outpatient department till discharge from hospital (mean per patient) has to be minimized.
- Number of outpatients admitted in the outpatient department, that have not finished at end of day (mean per week) has to be minimized.

The three key figures mentioned are equally weighted in the chromosome fitness function.

Out of these figures the dedicated additional fitness for each chromosome is calculated. This fitness rating is only used to calculate the specific mutation rate for each chromosome of an individual when generating new individuals.

To calculate the specific mutation rate, a linear function is used, which Kühn et al. [20] have seen as the best choice. The mutation rate can vary between 0.5 in maximum and 0.001 in minimum. This was chosen here, without any testing upfront. 0.5 seems the maximum mutation rate we can imagine, that not ends up in random search, and 0.001 is the minimum chosen mutation rate found in the literature. The control

strategy is adjusting the mutation rate within these borders by itself, relating to the chromosome fitness. When the chromosome fitness increases, the mutation rate decreases. So, at the end of an evolution run the mutation rate should be low. But, it seems possible that one or more chromosomes might end up with high mutation rates. A reason can be, e.g. that just very view patient admissions are planned. This might lead to bad fitness rating for single chromosomes even though it is the best solution in the overall view. To prevent high mutation rates at chromosomes, especially at the end of optimization run and to make sure that the minimum mutation rate of 0.001 will be reached at the end of a given run length, the approach of Kühn et al. [20] was expanded by implementing an additional deterministic rule. That lowers the maximum possible mutation rate over time.

So far, just the additional chromosome fitness and how to calculate the chromosome specific mutation rates out of it was explained. Selection is done based on the individual fitness, by fitness proportional selection (roulette-wheel). To measure the quality of an individual (solution) a cost function is implemented, calculating an earnings-cost value for each individual.

$$EC(i) = \sum_{i=1}^{n}(Earnings_i) - \sum_{i=1}^{n}(Costs_i);\ n \in N \quad (8)$$

The resulting value is representing the main optimization objectives. Earnings are calculated as follows:

$$Earnings = \mathrm{InPat} * 60 + \mathrm{HCL} * 500 + \mathrm{OutPat} * 60 + \mathrm{ECG} * 30 + \mathrm{ECHO} * 40 \quad (9)$$

InPat : number of elective patient admission at ward
OutPat: number of outpatient admission at ward at ward
HCL: number of HCL procedures carried out
ECG: number of ECG examinations carried out at patients from outpatient department
ECHO: number of ECHO examinations carried out at patients from outpatient department

ECG and ECHO are not considered as earnings for inpatients, following the given fact in the hospitals.

Costs are calculated as follows:

$$Costs = \mathrm{DIV(WT/15)} * -2 + \mathrm{WT} * 1 + \mathrm{InLoS} * 1 + \mathrm{OutLoS} * 1 + \mathrm{OutPat}_{inc} * 50 + \mathrm{InPat}_{inc} * 50 + \mathrm{HCL}_{TS} * 100 \quad (10)$$

WT: patients waiting time at all during the whole stay
InLoS: inpatients length of stay at all in days
OutLoS: outpatient length of stay at all in hours
$OutPat_{inc}$: number of patients in the outpatient department that have not left at the end of day
$InPat_{inc}$: number of patients on ward waiting for a bed over night after admission
HCL_{TS}: number of procedures in HCL, that have not been carried out at day planned

The resulting value has to be maximized. To calculate the individual fitness out of the earnings-cost value a ranking followed by a linear fitness assignment (linear ranking) was used. The sorted list of individuals is ranked, while the best individual is getting rank 1. The fitness *F* is calculated in regard to the rank *i* and population size *n* by:

$$F(i) = \frac{2 * (n + 1 - i)}{n * (n + 1)} \quad (11)$$

For optimization purpose the dynamic simulation model provided was extended to an optimization model, with attached GA. Before any of the experiments described in the following were done, an extensive final validation was done.

5. EXPERIMENTAL DESIGN

To optimize the chosen problem of real-world and to analyze the effects of the adaptive mutation rate, several test scenarios were defined. The optimization follows in a loop, as shown in Figure 2. The loop will be done as long as the optimization algorithm (GA) is running. The run length of GA is defined by the number of generations to be calculated.

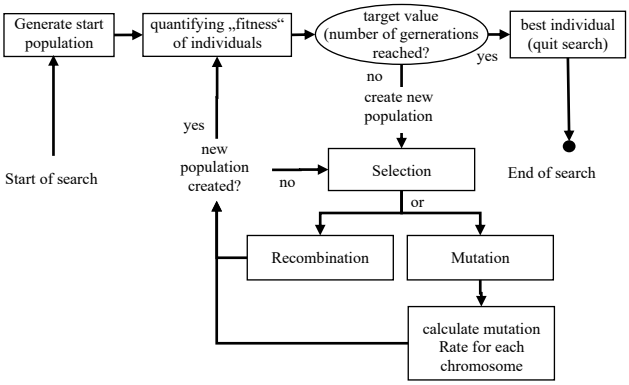

Figure 2. General sequence of actions for the genetic algorithm (based on [52, p. 9])

Following Kühn et al. [20] after selection either recombination OR mutation will be carried out to create new individuals. The reason is, when doing recombination first, in that moment a new individual is created and the upfront calculated chromosome fitness nor the calculated mutation rate fits to that new created individual. Furthermore, either recombination or mutation is performed for reproduction, to validate the Approach of Kühn et al. [20] on the chosen complex real-world problem.

The experimental design is based on two test scenarios (see Table 4). In scenario 1 the mutation rate (p_m) is non-adaptive during the optimization run and the same for every chromosome. Each variant of mutation rate in scenario 1 and 2 will be tested in separate evolutionary runs on a crossover probability (p_c) of 0.60 and 0.75, following De Jong [3] and Schaffer et al. [4]. For every variant, the run length of 200, 225 and 250 generations will be tested. In some runs during implementing and testing the GA, it was seen that this is a suitable range of run lengths. Over all, this leads to 54 optimization runs in scenario 1 (6 runs for each variant) and 6 runs in scenario 2. For every test scenario, the optimization model and all of its parameters will be kept unchanged, except the parameters shown in table 4.

Table 4: Test scenarios with variants

Mutation rate (p_m)	**Crossover probability** (p_c)	**Number of generations** (run length)
non-adaptive mutation rates 0.001, 0.005, 0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.5	0.60	200
		225
	0.75	250
adaptive mutation rate based on chromosome	0.60	200
		225

fitness (0.5 – 0.001)	0.75	250

The initial population is set up by random and contains 30 individuals. Elitism is not used (non-overlapping generations). As De Jong [2, p. 11] said, the "more 'elitist' a selection algorithm is, the more an EA behaves like a local search procedure (i.e., a hill climber, […]) and is less likely to converge to a global optimum". The fitness value is calculated with a linear ranking function and selection is done by roulette wheel strategy (see section 4). Crossover is done by one-point crossover at each chromosome. Pierrot & Hinterding [28, p. 144] have shown for multi-chromosome representations, that mutating only one chromosome did not give good results. They recommend to mutate one variable per chromosome instead of the average of one mutated variable per chromosome.

For every optimization run, at the end of the defined run time (number of Generations), the best Earning-Cost-Value (EC) reached is stored (as GA Performance). To minimize the stochastic influence each individual (solution) is tested 10 times with differing sequence of random numbers. The mean value is used, following [27, p. 117]. Based on this, scenario 1 and 2 are compared with each other. De Jong [2, p. 6] says that comparing non-adaptive parameter settings to one with adaptive settings is unfair since the non-adaptive settings were established via some preliminary parameter tuning runs. Keeping that in mind, the scenario 1 is compared to scenario 2 based on the GA performance.

For simulation purpose, MLDesigner v. 3.0 was used as simulation system. Each optimization run needs approx. 12 days in cpu time (Fujitsu Primergy TX200 S5, 2 x Intel® XEON® E5570 2,93 GHz Quad Core, 32 GB RAM). Because of that much optimization runs to be tested (~640 days in CPU time), a pool of 9 machines was used.

6. EXPERIMENTAL RESULTS

The two test scenarios were simulated and the effects on GA performance are measured. The following Table 5 shows the mean optimization results over all, Figure 3 visualizes the results.

Table 5: Resulting earnings-cost-value of scenarios 1 + 2

scenario	mutation rate	earnings-cost-value (CR 0.6 and CR 0.75) N = 60		
		best	worst	mean
scenario 1: non-adaptive mutation rate	0.001	172,180	165,919	168,564.33
	0.005	196,337	165,815	184,423.83
	0.01	197,022	186,331	191,381.33
	0.02	197,298	182,547	**192,411.67**
	0.03	195,335	184,506	189,080.67
	0.05	194,076	180,315	188,458.50
	0.1	191,464	174,176	181,274.50
	0.2	185,164	168,802	177,164.50
	0.5	176,703	161,731	168,458.33
scenario 2: adaptive mutation rate	0.5 – 0.001	198,272	188,250	**192,503.83**

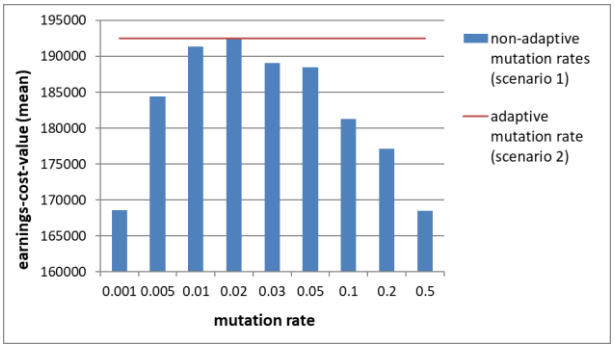

Figure 3. Resulting earnings-cost-value of scenarios 1 + 2

It is distinguished in the way mutation is done (scenario 1+2) and between the mutation rates applied. Each variant consists of 6 evolutionary runs (60 overall). At this moment, it is not distinguished by the crossover probabilities tested, so both variants of cross-over probability tested are included.

In both scenarios, it was aimed to maximize the earning-cost value (EC). In scenario 1, using non-adaptive mutation rates, the best mean result was 192,412 and was reached when applying a mutation rate of 0.02 (see Table 5). For scenario 2 with adaptive mutation rate, it is close to the same, with best mean value of 192,504 (see Table 5). On a significance level of 0.05, there was no statistic significant difference in mean values of these two variants. Thus, the adaptive parameter control delivers comparable results to the best non-adaptive mutation rate, without preliminary parameter tuning. That makes adaptive parameter more efficient and upfront hand tuning obsolete.

Looking at scenario 1, the non-adaptive mutation rates less than 0.005 and higher than 0.1 have shown a quite poor GA performance. This behavior was anticipated and caused by too low (insufficient diversity – clones appear) and at > 0.1 too high (highly destructive) mutation rates. On the chosen complex optimization problem, best GA performance (mean) can be reached at a mutation rate of 0.02. Compared to former work, like De Jong [3], Schaffer et al. [4] or Grefenstette [15], the best mutation rate found here appears to be a little higher. At this point it is not clear, what causes that. It can be a result of the fact that in our test case a multi-chromosomal representation is used, that might need a higher mutation rate. A reason can also be, that either mutation or crossover was used to create new individuals (see experimental design in section 5).

The following Table 6 and Table 7 are presenting the earnings-cost-values for scenario 1+2 separated by the applied crossover probability of 0.60 and 0.75.

As it can be seen, the resulting GA performance depends also on the crossover probability. For crossover probability of 0.6 the adaptive mutation rate performs not that good as using a crossover probability of 0.75 on the chosen complex optimization problem. A reason can also be seen in the limited number of GA runs.

Table 6: Resulting earnings-cost-value of scenarios 1 + 2 at CR 0.6

scenario	mutation rate	earnings-cost-value (CR 0.6) N = 30		
		best	worst	mean
scenario 1: non-adaptive mutation rate	0.001	165,919	169,913	168,461.00
	0.005	190,885	196,337	193,578.33
	0.01	189,446	195,444	191,497.33
	0.02	193,059	196,079	**194,779.00**
	0.03	184,506	192,690	187,811.00
	0.05	180,315	186,486	183,940.33
	0.1	174,176	178,250	176,021.67
	0.2	168,802	174,747	171,797.67
	0.5	161,731	164,535	163,399.67
scenario 2: adaptive mutation rate	0,5 – 0,001	188,250	191,101	**189,576.67**

Table 7: Resulting earnings-cost-value of scenarios 1 + 2 at CR 0.75

scenario	mutation rate	earnings-cost-value (CR 0.75) N = 30		
		best	worst	mean
scenario 1: non-adaptive mutation rate	0.001	172,180	166,050	168,667.67
	0.005	182,195	165,815	175,269.33
	0.01	197,022	186,331	191,265.33
	0.02	197,298	182,547	190,044.33
	0.03	195,335	187,070	190,350.33
	0.05	194,076	191,963	**192,976.67**
	0.1	191,464	183,234	186,527.33
	0.2	185,164	179,082	182,531.33
	0.5	176,703	170,699	173,517.00
scenario 2: adaptive mutation rate	0,5 – 0,001	198,272	193,975	**195,431.00**

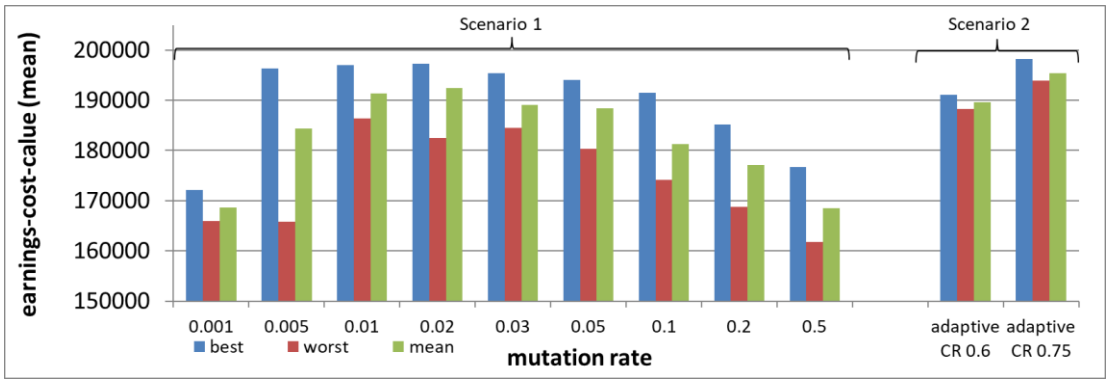

Figure 4. Results separated by best, worst and mean results

Figure 4 shows the best (blue), worst (red) and mean (green) results reached for both scenarios, separated by mutation rate and the way mutation is done (scenario 1+2). The Results of scenario 1, can be seen on the left side, separated by the non-adaptive mutation rates (0.001 – 0.5). For scenario 1, both variants of crossover probability are included (mean). Scenario 2 is shown on the right side, separated by the crossover probabilities (CP).

In general, a quite good GA performance can be seen in scenario 2, independent of the crossover probability. The adaptive mutation rate, calculated by chromosome fitness, performed a little better at a crossover probability of 0.75 than at one of 0.60. The same can be observed for non-adaptive mutation rates. Further on, the results of adaptive mutation rate seem quite robust, as the best, worst and mean values are close together, compared to non-adaptive variants. But, maybe there are more runs needed to say this in general. Like it was said before, a lot of CPU time is needed for testing. And it was not aimed to statistically prove the approach itself. The outperformance of this approach in general was statistically proven upfront with a huge number of runs on a set of theoretical test functions in Kühn et al. [20].

The following Figure 5 visualizes the adaptive mutation rates applied to the chromosomes CR01 – CR06 over time. These chromosomes are designed to plan reservations, sequencing and scheduling of elective patients in ECG, ECHOs and Op-theaters.

Figure 6 shows the earnings values over time for one variant of adaptive mutation rate with CR of 0.75. Each dot is a tested solution (mean value simulated 10 times). It can be seen that at the beginning better solutions can be found very fast. With growing number of generations passed, it takes more time to find better solutions (convergence of GA). By the spreading of solutions it can be seen, how adaptive mutation rate is generating more solutions with low earnings value at the beginning of the optimization run and less at the end, when mutation rate is lower. Figure 7 shows an example when static mutation rate of 0.02 and CR of 0.75 is used instead.

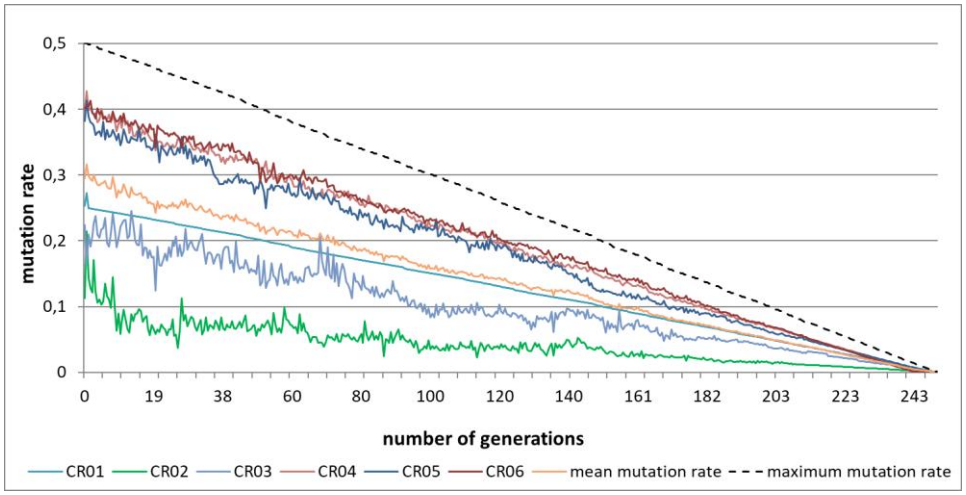

Figure 5. Applied adaptive mutation rates over time, separated by chromosomes

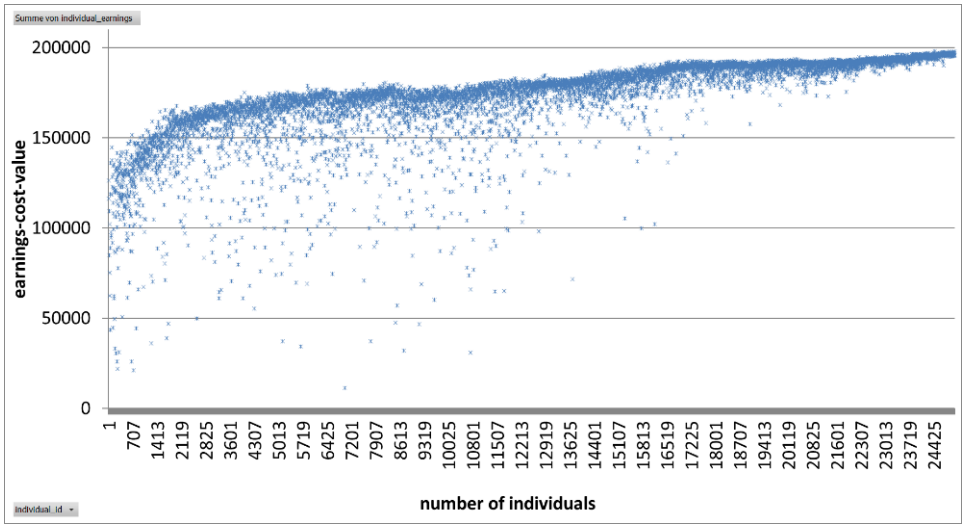

Figure 6. Individuals optimized by adaptive mutation rate over time

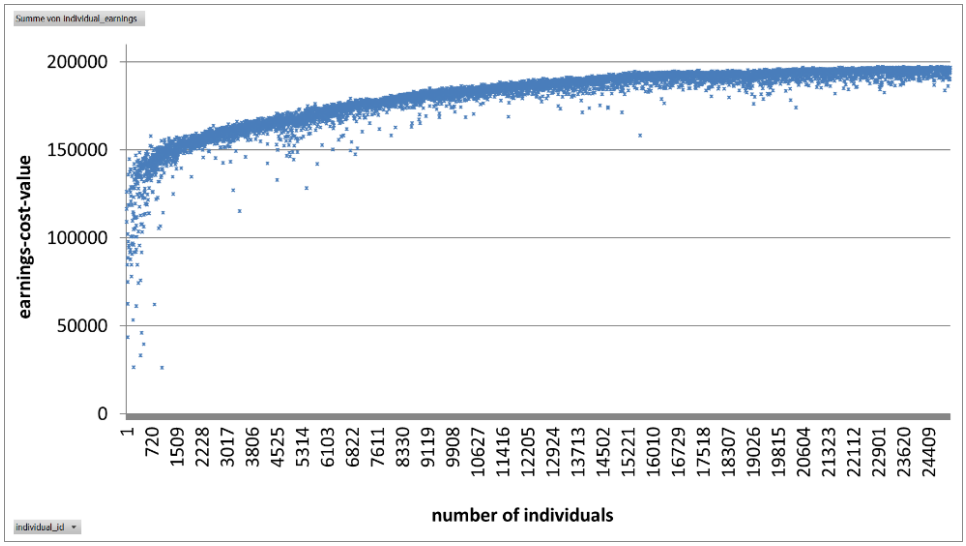

Figure 7. Individuals optimized by non-adaptive mutation rate of 0.02 over time

7. SUMMARY AND OUTLOOK

It was shown, on a complex real-world optimization problem, how dynamic coupled system of systems (SoS) can be optimized by using a Genetic Algorithm (GA). We used a dynamic, executable simulation model, which was very detailed and considers the uncertainty of real-world processes, to optimize outpatient admission, inpatient admission and op-theater planning simultaneously for the first time. We used dedicated chromosomes to represent (sub-)systems (multi-chromosome representation), with their individual objectives and constraints of planning. Thus, we were able to optimize and measure specific criteria dedicated to each chromosome (system).

To increase the GA performance our approach uses an adaptive mutation rate. We considered aspects of diversity and fitness together to make the mutation rate adaptive. While creating new individuals every chromosome has its dedicated mutation rate, depending on its goodness. Thereby, we see two level of hierarchy. The specific objectives of (sub-)systems and the main objectives for the system as a whole. Both were optimized together.

For the complex real-world optimization problem, preliminary parameter tuning runs were performed, to figure out the non-adaptive mutation rate, where the best optimization results can be reached within a given period of time (number of generations). We have shown, that optimization results with the best non-adaptive mutation rate is comparable to our GA with adaptive mutation rate. That confirms the upfront tests on a set of test functions and shows that the expected results can also be achieved when applying our approach to the chosen complex problem of real world.

It was not our goal to achieve better results on applying adaptive mutation rate compared to non-adaptive. The main improvement can be seen in overcoming the need to hand tune control parameters upfront. This is important and very time saving.

Most of the research in the field of EA has been done on a theoretical basis. Often the proposed solutions do not deliver what they promise, when applying them to complex real-world problems. Thus, to confirm research, more tests on complex optimization problems of real world are needed, especially for multi-objectives.

Besides mutation rate, also other control parameters should be adaptive, e.g. crossover probability and populations size. Like De Jong [2, p.15] said, “the ultimate goal of these efforts is to produce an effective and general problem-solving EA with no externally visible parameters.” This will only be achieved if there are effective ways to dynamically adapt various internal parameters.

8. REFERENCES

[1] Holland, J. H. 1975. Adaption in natural and artificial systems, Ann Arbor, Michigan, USA: Univ. of Michigan Press.

[2] De Jong, K. A. “Parameter setting in EAs: a 30 year perspective”. In Lobo et al. [33], pp. 1–18.

[3] De Jong, K. A. 1975. An analysis of the behavior of a class of genetic adaptive, Ph.D. thesis, University of Michigan.

[4] Schaffer, J. D., Caruana, R. A., Eshelman, L. J. and Das, R. “A study of control parameters affecting online performance of genetic algorithms for function optimization”. In Proceedings of 3rd Int. Conf. Genetic Algorithms, 1989, pp. 51–60.

[5] Eiben, A. E. and Smith, J. E. (2015). Introduction to evolutionary computing. 2nd ed. (Natural Computing Series), Berlin, Heidelberg, Germany: Springer, doi: 10.1007/978-3-662-05094-1.

[6] Eiben, A. E., Hinterding, R. and. Michalewicz, Z. “Parameter control in evolutionary algorithms”. IEEE Trans. on Evolutionary Computation, 3(2), 1999, pp 124–141.

[7] Bäck, T. 1996. Evolutionary algorithms in theory and practice. Oxford. UK: Oxford University Press.

[8] Reeves, C. “Using Genetic Algorithms with small Populations”. In Procedings of 5th Int. Conf. Genetic Algorithms, 1993, pp. 92–99.

[9] Rechenberg, I. 1973. Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart. Germany: Frommann-Holzboog.

[10] Schwefel, H.-P. 1981. Numerical Optimization of Computer Models. New York. USA: Wiley.

[11] Beyer, H.-G. and Schwefel, H.-P. “Evolution strategies - A comprehensive introduction”. In Natural Comput, 2002, vol. 1. no.1, pp. 3-52.

[12] Eiben, A. E., Michalewicz, Z., Schoenauer, M. and Smith, J. E. “Parameter control in Evolutionary Algorithms”. In Lobo et al. [33], pp. 19-46.

[13] Fogarty, T. C., “Varying the probability of mutation in the genetic algorithm”. In Proceedings of 3rd Int. Conf. Genetic Algorithms. 1989, pp. 104–109.

[14] Hesser, J. and Männer, R. “Towards an Optimal Mutation Probability for Genetic Algorithms”, Parallel Problem Solving from Nature (Lecture Notes in Computer Science vol. 496). Berlin, Heidelberg. Germany: Springer. 2005, ch. 4, pp. 23–32, doi: 10.1007/Bfb0029727.

[15] Grefenstette, J. J. “Optimization of control parameters for genetic algorithms”. In IEEE Trans. Syst., Man, Cybern., vol. 16, no. 1, (Jan. 1986), pp. 122–128, doi: 10.1109/TSMC.1986.289288.

[16] Bäck, T. 1992. Self-Adaption in Genetic Algorithms. In Proceedings of 1st European Conf. Artificial Life, pp. 263–271.

[17] Bäck, T. “Optimal mutation rates in genetic search”. In Proceedings of 5th Int. Conf. Genetic Algorithms, 1993, pp. 2–8.

[18] Smith, J. “Parameter perturbation mechanism in binary coded gas with self-adaptive mutation”. 7th Int. Workshop FOGA, 2003, pp. 329-346.

[19] Hinterding, R. “Self-adaptation using multi-chromosomes”. In Proceedings of IEEE Int. Conf. Evol. Comput., 1997, pp. 87–91.

[20] Kühn, M., Severin, T. and Salzwedel, H. “Variable Mutation Rate at Genetic Algorithms: Introduction of Chromosome Fitness in Connection with Multi-Chromosome Representation”. Int. Journal Comput. Appl., vol. 72, no. 17, 2013, pp. 31–38.

[21] Lippold, J. 2014. Aufbau eines prozessorientierten Simulationsmodells für klinische Einrichtungen zur abteilungsübergreifenden Termin- und Reihenfolgeplanung. Dipl.-thesis, Tec. Univ. Ilmenau.

[22] Gonçalves, J. F., de Magalhães, M. J. J. and Resende, M. G. C. 2002. A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem. AT&T Labs Research. Technical Report TD-5EAL6J.

[23] Keedwell, E. and Khu, S.-T. "A hybrid genetic algorithm for the design of water distribution networks". Engineer Appl. of Artificial Intell., vol. 18, no. 4, Jun. 2005, pp. 461–472, doi:10.1016/j.engappai.2004.10.001.

[24] Kühn, M., Baumann, T. and. Salzwedel, H. "Genetic Algorithm for process optimization in hospitals". In Proceedings of 26th Eur. Conf. Modelling Simulation, 2012, pp. 103–107.

[25] Gao, W. "Study on New Improved Hybrid Genetic Algorithm". Advances in Information Technology and Industry Applications (Lecture Notes in Electrical Engineering vol. 136), ch. 66, 2012, pp. 505–512 doi: 10.1007/978-3-642-26001-8_66.

[26] Davidor, Y. "Genetic Algorithms and Robotics - A Heuristic Strategy for Optimization" World Scientific Series in Robotics and Intelligent Systems vol.1, 1991. Singapore: World Scientific.

[27] Juliff, K. "A multi-chromosome genetic algorithm for pallet loading". In Proceedings of 5th Int. Conf. Genetic Algorithms 1993, pp. 467–473.

[28] Pierrot, H. J. and Hinterding, R. "Using multi-chromosomes to solve a simple mixed integer problem". Lecture Notes in Computer Science, vol. 1342, 1997, pp. 137–146.

[29] Ronald, S., Kirby, S. and Eklund, P. "Multi-Chromosome Mixed Encodings for Heterogenous Problems". In Proceedings of 4th Int. Conf. Evol. Comput., 1997, pp. 37–42.

[30] Wight, J. and Zhang, Y. "An 'Ageing' Operator and Its Use in the Highly Constrained Topological Optimization of HVAC System Design". In Proceedings of 2005 GECCO, pp. 2075–2082.

[31] Cavill, R., Smith, S. and Tyrrell, A. "Multi-Chromosomal Genetic Programming". In Proceedings of 2005 GECCO, pp. 1753–1759.

[32] Kerati, S., Moudani, W. E. L., de Coligny, M. and Mora-Camino, F. "A Heuristic Genetic Algorithm Approach for the Airline Crew Scheduling Problem". In IEEE Congr. Evol. Comput., 2009, pp. 1383–1390.

[33] Peng, J. and Chu, Z. S. "A Hybrid Multi-Chromosome Genetic Algorithm for the Cutting Stock Problem". In ICIII, 2010, pp. 508–511.

[34] Lobo, F. G., Lima, C. F. and Michalewicz, Z. (Ed.) "Parameter Setting in Evolutionary Algorithms". In Studies in Computational vol. 54, Berlin, Heidelberg, Germany: Springer, 2007.

[35] Meyer-Nieberg, S. and Beyer, H.-G. "Self-Adaptation in Evolutionary Algorithms", in Lobo et al. [33], pp. 47–75.

[36] Eiben, A. E. and Smit, S. K. "Parameter tuning for configuring and analysing evolutionary algorithms", Swarm, Evol. Comput., vol. 1, no. 1, pp. 19–31, Mar. 2011, doi: DOI: 10.1016/j.swevo.2011.02.001.

[37] Karafotias, G. H. M. and Eiben, A. E. "Parameter Control in Evolutionary Algorithms: Trends and Challenges". IEEE Trans. Evol. Comput., vol. 19, no. 2, Apr. 2015, pp. 167-187, doi: 10.1109/TEVC.2014.2308294.

[38] Fernandes, C. M., Merelo, J. J., Ramos, V. and Rosa, A.C. "A Selforganized criticality mutation operator for dynamic optimization problems". In Proc. 2008 GECOO, pp. 937–944

[39] Severin, T. 2014. Implementierung eines Genetischen Algorithmus mit Multichromosomenansatz zur abteilungsübergreifenden Termin- und Reihenfolge-planung in Kliniken", Dipl.-thesis, Tec. Univ. Ilmenau.

[40] Aleti, A. and Moser, I. "A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms". ACM Computing Survey CSUR, vol. 49, no. 3, Article 56, 2016.

[41] Eiben, A. E., Hinterding, R. and Michalewicz, Z. „Parameter Control in Evolutionary Algorithms". IEEE Trans. Evol. Comput., vol. 3, no. 2, 1998, pp. 124–141.

[42] Serpell, M. and Smith, J. E. "Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. J. Evol. Comput., vol. 18, no. 3, Sep. 2010, pp. 491-514, doi: 10.1162/EVCO_a_00006.

[43] Yuan, B. and. Gallagher, M. "Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks", in Lobo et al. [33], pp. 121-142.

[44] Kühn, M., Lippold, J. and Salzwedel, H. "Automatic transformation of hospital processes into executable model with EPML". Int. J. Comput. Appl., vol. 80, no. 9, 2013, pp. 20–30.

[45] Page, B. 1991. Diskrete Simulation: Eine Einführung mit Modula-2 (Springer Lehrbuch). Berlin. Heidelberg. Germany: Springer.

[46] Cayirli, T., Veral, E. "Outpatient scheduling in health care: a review of literature". Prod., Operations Manage., vol. 12, no. 4, pp. 519–549, doi: 10.1007/978-3-662-05094-1, Jan. 2009.

[47] Gupta, D., Wang, W.-Y. "Patient Appointments in ambulatory Care". Handbook of Healthcare System Scheduling (Int. Series in Operations Research & Management vol. 168), R. Hall, Ed. Boston, MA, USA: Springer, ch. 4, pp. 65–104, doi: 10.1007/978-1-4614-173-7_4, Nov. 2011.

[48] Cardoen, B., Demeulemeester, E. and Beliën, J. "Operating room planning and scheduling: A literature review", Eur. J. Oper. Res., vol. 201, no. 3, pp. 921–932, doi: 10.1016/j.ejor.2009.04.011, Mar. 2010.

[49] Helm, J. E., Lapp, M., See, B. D. "Characterizing an effective hospital admissions scheduling and control management system: A genetic algorithm approach". In Proceedings of 2010 WSC, 2010, pp. 2387–2398.

[50] Gemmel, P., van Dierdonck, R. "Admission scheduling in acute care hospitals: does the practice fit with the

theory?". Int. J. Operations, Prod. Manage., vol. 19, no. 9, Sep. 1999, pp. 863–871, doi: 10.1108/01443579910280188.

[51] Nissen, V. and. Biethahn, J. „Ein Beispiel zur stochastischen Optimierung mittels Simulation und einem Genetischen Algorithmus". In Simulation als betriebliche Entscheidungshilfe. State of the Art und neuere Entwicklungen, Biethahn, J., Hummeltenberg, W., Schmidt, B., Stähly P., and Witte, TH. (Ed.) Heidelberg, Germany: Physica, 1999, ch. 6, pp. 108–125, doi: 10.1007/978-3-642-58671-2_6.

[52] Pohlheim, H. 2000. Evolutionäre Algorithmen. Verfahren, Operatoren und Hinweise für die Praxis, Berlin, Germany: Springer.

[53] Kühn, M., Baumann, T., Salzwedel, H. "Genetic algorithm for process optimization in hospitals". In Proceedings of 26th European Conference on Modeling and Simulation, 2012, pp. 103–107.