A Regulatory Algorithm (RGA) for Optimizing Examination Timetabling

Christina Klüver
University of Duisburg-Essen
Institute for Computer Science and Business Administration
Essen, Germany
christina.kluever@uni-due.de

Jürgen Klüver
University of Duisburg-Essen
Computer Based Analysis of Social Complexity
Essen, Germany
juergen.kluever@uni-due.de

***Abstract*—We describe the regulatory algorithm (RGA), a two dimensional extension of standard evolutionary algorithms. Its possibilities are shown by an application to the problem of optimizing timetabling for exams with real data from the University Duisburg-Essen (Germany). The results of the RGA application show that the room allocation problem for written exams can be satisfactory solved in a few minutes. In addition we compared the RGA with a standard GA. The RGA was significantly better in all experiments; in particular the GA could not fulfill all distribution demands in contrast to the RGA.**

Keywords—optimization, evolutionary algorithms, regulatory algorithm, genetic algorithm, constraints, room allocation plans

I. INTRODUCTION

The problem of optimizing room allocation plans or optimizing timetables is a well-known problem of the class of NP-complete problems. Especially so-called hard and soft constraints are a major challenge for the administrators and/or for computer programs accordingly. Because of its practical importance many approaches have been developed for its solution, for example graph theoretical ones, the usage of specific neural networks, evolutionary algorithms, and simulated annealing (cf. e.g. [1] - [7]).

A similar problem occurs for the problem of exam timetabling (cf. e.g. [8] – [12]). This task is even more difficult for the administration because it has only a little time to organize the rooms for all written exams during a short period.

In this article we describe the application of a new evolutionary algorithm to this problem, namely the Regulatory Algorithm (RGA) that was developed by our research group CoBASC[1] [13]. We obtained from the administration of our University Duisburg-Essen the complete data for one semester, e.g. number of lecturers, number of exams, number of students per examination, and applied the RGA to construct an optimal room allocation plan for the written exams. To demonstrate the thickness of the algorithm only the timetabling for student teachers are considered because they have the plan priority at the university.

The aim of the article is twofold: on the one hand we present the basic logic of RGA because it is still a relative new system; on the other the chief goal of the article is a demonstration how to solve an important practical and rather complex problem with the RGA. The obvious practical success of the RGA is an important indicator for its validity as a suited tool for solving complex optimization problems.

We first describe the RGA; in the following section the situation at the university is given; subsequently we show the RGA model for the room allocation problem for exams and describe the experimental design, and present the most important results. Finally we show the results of comparisons with a standard GA.

II. THE REGULATORY ALGORITHM (RGA)

The usual evolutionary algorithms like in particular Genetic Algorithms (GA), Evolution Strategies (ES), and Genetic Programming (GP) are, as is well known, biologically orientated to the *Modern Synthesis* [14], i.e. the integration of Darwin's theory of evolution and Mendelian genetics. There are, according to this classical paradigm, genes as the fundamental units of evolution and all genes are insofar from the same type as they determine the development of the different parts of the organism. According to this fundamental conception the evolutionary algorithms inspired by it are basically one-dimensional systems, namely strings or vectors respectively, consisting of components as the formal representatives of genes. To be sure, the components may consist themselves of different units as, e.g., in GP where the components consist of "trees", but the basic logic is always the same.

When applying evolutionary algorithms to a certain optimization problem it is usually represented, as we mentioned, in form of a vector or string respectively. Hence we have basically a one-dimensional representation, which can be visualized as follows (Fig. 1):

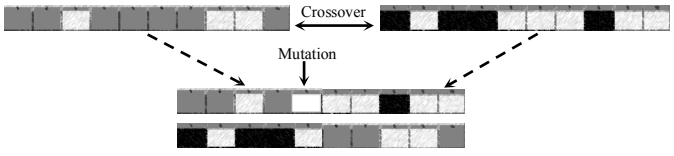

Fig. 1. A one-dimensional vector representation of two „individuals", upon which mutation and crossover perform the variation, determined by the according fitness or evaluation function

[1] Computer Based Analysis of Social Complexity

In terms borrowed from biology the vector frequently is called an "individual" and its components are the "genes". Hence a population of individuals is a set of such vectors.

At least since the fundamental work of Jacob and Monod [9] one has in biology to distinguish between different types of genes: The genes that are responsible for the development of the organism are no autonomous units but are dependent from *regulatory genes* that "switch on and off" the first genes (Jacob and Monod called these first genes *structural genes*). Therefore, the genome must be understood as a system that consists of different levels where each level consists of a particular type of genes with different functions (cf. e.g. [15]). The result of an orientation to these insights is the *Regulatory Algorithm (RGA)*.[2]

An "individual" that is constructed for the usage of a RGA is more complex because it must be a (at least) two-dimensional representation. It consists of a) a "regulatory vector" (RV), whose elements are "regulatory genes" (rg), b) of a second vector, namely the "structural vector" (SV), containing the "structural genes" (sg), and c) a set or vector of connections (CV) between the regulatory genes and the structural genes (Fig. 2). By using the terms "regulatory genes" and "structural genes" we take over the original terminology of Jacob and Monod.

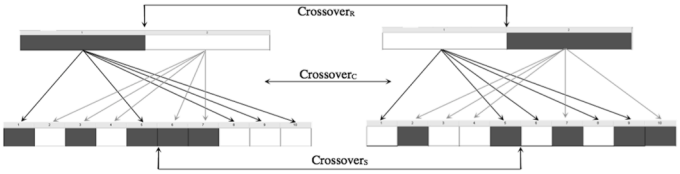

Fig. 2. Two RGA individuals constructed as two-dimensional representations; crossover is performed according to the arrows between the different parts of the individuals.

Fig. 2 should be understood the following way: In each individual the regulatory genes are connected with one or several structural genes but not vice versa. This means that the regulatory genes have effects on the structural genes, i.e. determine their values or their position in the vector, but the structural genes do not influence the regulatory genes. Hence an individual is a simple two-dimensional topological space whose topology is defined by the connections between the components of RV and SV. The kind of effects on the structural genes depends on the specific problem. In the simplest form the structural genes are just "switched on" and "switched off" by the regulatory genes, according to the biological terminology. If for example the regulatory genes are just binary coded then a regulatory gene in a state of 1 might leave the structural genes in their original state; if the regulatory gene is in a state of 0 then the connected structural genes will for example also become zero or they will be omitted from the fitness function and so on. There are several possibilities already in this simple case.

In the case of coding with real numbers other possibilities exist to define the effects of the regulatory genes on the structural genes, for example multiplying the state value of a regulatory gene with that of the connected structural genes. Which possibility a user chooses depends of course again on the specific problem. Finally one might also "weight" the connections, i.e. define each connection not only by its topological function but also by a specific weight value. In this case a RGA individual becomes similar to a two-level feed forward neural network; the effects of the regulatory genes on the connected structural genes might be computed in orientation to well-known activation functions in such neural nets.

In addition to such "numerical" effects the regulatory genes might have on the structural genes it is also possible to define the regulatory genes as "shifting orders", i.e. as rules for a recombination of the structural genes. Other possibilities can easily be imagined (for details cf. [13]).

According to the specific problems the fitness function could a) be applied to the whole individual or b) to specific components, for example the set of connections. Usually the evaluation of the whole individual would be the most natural form of evaluation in such cases. In our application only SV is evaluated, i.e. the fitness value of the individual is that of SV.

The application of the usual genetic operators, i.e. mutation and crossover, to a RGA individual can be done in seven different ways: (1) Only the RV is varied; (2) only the CV is varied; (3) only the SV is varied; (4) RV and CV are both varied; (5) RV and SV are both varied; (6) CV and SV are both varied; (7) all three parts are varied. Variation of two or more parts of a RGA individual means that in a predetermined order first one part is varied and then the next. If for example both RV and CV should be varied then a natural order would be first the variation of RV and then of CV; in this case of course the CV for the variation operations is defined by the result of the previous variation of RV.

On a first sight it would seem that possibility (3), namely just varying the SV, is the same as just using a GA (or an ES). Yet this is only the case if one blocks all other parts of the RGA individual, i.e. if there are no influences of RV and CV on SV. If all parts of the individual are still connected then a variation of the SV would connect its components with different elements of the RV, which frequently would change the values of the components of SV.

Factually things are even more complex. The seven combinatorial possibilities require that the genetic operators, i.e. crossover and mutation, are always applied to the according levels in the same manner. If for example option (4) is chosen, i.e. the variation of both RV and CV, and if both operators should be used on RV, then the same must be the case with the variation of CV. This restriction is of course not necessary. Other possibilities are, for example, to vary in this example RV with both operators and CV just with mutation. Hence, if we take into account these additional variation possibilities we get not only seven, which is complicated enough, but altogether $2^7 = 128$. In most practical applications, however, one of the seven basic options is sufficient.

When using a RGA for a specific task a *variation schema* VS = ((x,y), (x,y), (x,y)) can be defined as a pragmatic overall

[2] The idea to combine regulatory genes with Genetic Algorithms was also proposed e.g. by [17] - [19].

view; x = 1 and y = 0 for example in the first bracket means that crossover shall be applied to RV and mutation not; the second bracket refers the same way to the variation of CV and the third bracket refers to the possible variation of SV. When taking, e.g., the option (4) with the variation just described we have

$$VS = ((1,1),(0,1),(0,0)), \quad (1)$$

namely both operators shall be applied to the RV, only mutation to CV, and no operator to SV.

The variation schema of the basic option (7), i.e. varying all components of RGA individuals with both operators would accordingly be

$$VS = ((1,1), (1,1), (1,1)). \quad (2)$$

According to the seven basic options of varying RGA individuals there are also seven basic options to introduce elitist versions of the RGA, namely taking over into the next generation from the best individual(s) (1) the RV, (2) the CV, (3) the SV, (4) RV and CV, (5) RV and SV, (6) CV and SV, and finally (7) the whole individual.

Without doubt the RGA is a much more complex algorithm than, e.g. a GA or an ES. Therefore, the question suggests itself why one should use such a complex algorithm if simpler ones are at one's disposal, which have been already analyzed for several decades. In our opinion the usage of a RGA might have several advantages:

(a) The very different sizes of the RV and the SV, as found in biological experiments, suggest the possibility that at least sometimes biological evolution might have operated chiefly on the RVs and, because a RV is much smaller than the according SV, could perform significantly faster. Hence a RGA might be faster than a GA or an ES if only the RV is varied. This must not be necessarily true with all problems but the possibility is rather promising. Indeed, in some cases, which we analyzed [cf. 7], a RGA was significantly faster than a usual GA, whose performance we compared with the performance of the RGA.

(b) It is well known that in many problems like the TSP specific constraints must be taken into account. The difficulty is that in particular the crossover operator frequently generates "wrong" individuals, for example in case of the TSP routes where one town might appear twice and other towns not at all. Already Michalewicz [20] described three classical ways of dealing with this problem, namely the "penalty approach", the "decoder approach", and the "repair approach". As far as we know, the penalty approach, namely the "punishment" of wrong individuals by reducing their values in the fitness function, and the repair approach, namely the introduction of additional rules for removing wrong individuals from the population, are the mostly used techniques. But both approaches are usually very problem specific and not very elegant, as already Michalewicz observed.

By using a RGA problems like TSP can be dealt with without any additional rules. The RV is in such cases defined as a "shifting order", namely as an instruction how to recombine the structure genes; only the RV is varied by mutation and crossover. Because there are no variations of the SV wrong combinations in the sense mentioned above cannot appear and hence no additional rules like penalty or repair ones are necessary.

(c) The third possibility is the modeling and optimizing of many leveled systems. Consider for example a social organization with at least two different levels, namely superiors (= RV) and employees (= SV). Obviously such a system could very easily be represented by a RGA; if this system should be optimized according to certain criteria it is "just" a question of the practical possibilities of variation the system allows, which of the RGA components should be varied by the genetic operators. If, for example, only the level of employees can in reality be varied then the RGA must take this into account; if it is possible to change the relations between superiors and employees the connection vector CV should be varied, and if the superiors might be changed, i.e. their position on their level, RV is open to variations.

To be sure, such an organization frequently consists of more than two levels – three, four or even more. Yet as already Jacob and Monod, as we mentioned, proposed a three level model, it is easily possible to extend the RGA to three or more levels. In such cases the initial RV would become a "second level" SV with a "second level" RV above him and so on. Of course, the connections between the different levels would become more complicated but in any case the modeling of such multi level systems would be much easier than the usage of a GA or an ES *and* the construction of a suited model. In particular, if one has already a RGA shell at one's disposal with the possibility to increase the number of levels, as we have, the mapping of systems like social hierarchically structured systems on an according RGA would be a very natural way of modeling and a rather easy one too.[3]

To be sure, at present it is not possible to define general ways, which of the different possibilities should be used when dealing with a specific problem. Yet for example the problem of room allocations for written exams, as we shall show, demonstrates that a variation of RV is not useful and hence a variation schema VS = ((0,0), (x,y), (x,y)) should be applied. According to our experience the structure of the specific problem usually determines a suited VS.

III. Situation at the University Duisburg-Essen

The University Duisburg-Essen with approx. 36.000 students consists of the Campus Duisburg and the Campus Essen with different courses of study; for example mathematics and physics are offered at Campus Duisburg, medicine, chemistry, and biology at Campus Essen. Essen and Duisburg are two large towns in the *Ruhrgebiet*, an industrial region in the western part of Germany. Each campus is separately

[3] In the last years a new approach to use evolutionary algorithms has been proposed, namely the usage of so-called Multi Level Evolutionary Algorithms (MLEA) or more specifically Multi Level Genetic Algorithms (MLGA) (e.g. [21] and [22]). Despite the striking semantical similarity this approach is factually an inverse way: In MLEA a problem is decomposed in different "levels" and on each level an EA is applied. The algorithms remain one-dimensional in contrast to the RGA where the algorithm itself consists of different levels.

administered although in some courses students have to travel from one town to the other; the distance is about 20 km. The students have two time periods for the examinations, at the end of the term, and before the beginning of the next semester. A particular problem is that the teacher education students have lectures at both locations, and the exams take place accordingly. In addition, the exams have to be finished during a time period of two weeks; after this period the other exams, according to the number of students, have to be planed.

In total only a few weeks are at disposal to finish the first examination period. All students must have the results of their exams before the beginning of the registration time for the second time period, and they generally have the possibility to cancel the examination one week before the fixed examination date.

The lecturers have to inform the administration in a long lead time about the expected number of students, the requested day, the duration time of the examination, and needed special rooms like PC- or lab-rooms. In addition, many lecturers wish a factor for the space between two students to avoid cheating: e.g. factor 4 means that according four free places must be available between two students. This has of course an influence on the needed capacity: if 300 students are expected, and the space factor is 4 the administration has to reserve a location with a capacity for 1,200 students. Additional problems are if the administration has to rent a location outside of the university and that only one week before the examination date a lot of students cancel the exam registration.

The following Table I shows the dimension of the whole problem:

TABLE I. OVERVIEW OF THE EXAM ANNOUNCEMENTS

Exams for teacher education students	**NR.**
Subjects	261
Exam registrations	14,273
Space factor	1 – 4
Students per exam	1 – 1,795
Required room capacity (with a space factor)	4 – 5,000
Duration time of the exams	1 – 10 hours

An additional serious problem in practice and for the RGA is the fact that the requests for rooms with a specific capacity are very unevenly distributed. For example, the numbers of the expected teacher education students vary from 1 to 1,795 with a different factor for the free space between the students.

The problem, hence, for each day was to allocate suitable rooms that fulfill all or the most important demands for every announced request, i.e. the exams with specific wishes of day, time, and so forth.

We applied the RGA only to the allocation problem of the Campus Essen to demonstrate its possibilities.

IV. THE RGA MODEL

Each room suitable for a specific request must fulfill the conditions that characterize the request as nearly as possible. For the RGA model we defined these conditions as constraints that determine the success of the proposals of the RGA. A violation of a constraint, e.g. if the proposed room is too small for the expected number of students, will be "punished", i.e. the room proposal by the RGA receives an unfavorable fitness value (see below). Because not all requests of the exams are of equal importance we divided the whole set of constraints into "hard constraints" and "soft constraints". For example, the expected number of students is a hard constraint because obviously an exam in a room too small cannot be properly carried through; the request for a specific room is defined as a soft constraint because it is not very important, with the exceptions of PC- or lab-rooms. The violations of hard constraints are punished more severely than a violation of soft constraints. The constraints taken into account in the model are:

Hard constraints:

(1) *One exam one time*: One specific exam could be announced only for one specific time at a certain day.

(2) *One exam one room*: At a specific time only one *exam* might be delivered in a certain room.

(3) *No double exams*: A specific *exam* must be planned only once at a specific time. This means a consistency check for all exams.

(4) *Capacity*: Each room proposed by the RGA must contain sufficient space, i.e. chairs and tables, for the number of students.

(5) *All exams*: A proposed plan for room allocation must contain all *exams* announced by the lecturers. This is an additional consistency check.

Soft constraints:

(6) *Desired day*: The day wished by the lecturer.

(7) *Desired time start and necessary end*: The starting time wished by the lecturer.

(8) *Desired room*: A specific room wished by the lecturer.

It must be noted, however, that the number of rooms in the university is never sufficient for the written exams. That is why the university has to rent additional rooms outside, for example in the halls of the Essen fair. These additional rooms are also taken into the database of the RGA. The wish of the administration not to rent additional rooms or at least as few as possible outside of the university can be taken into account too. A proposal for an according enlargement of the RGA model is given at the end of this article.

A violation of one of the hard constraints will be punished by a value of 1.0, a violation of a soft constraint by values from 0.125 to 0.9, depending on the importance of the constraint.

In the RGA model the 8 constraints are implemented as regulatory genes (rg). The RV, hence, is a 8-dimensional vector RV = (con_1, con_2, …, con_8).

The requests are implemented as structural genes sg. Each gene consists of the components (requested) time, capacity, day, lecturer, and exam; the RGA substitutes for "capacity" a specific room with characteristics of its capacity, the factual equipment, and possibly if it is a room explicitly wished by the lecturer. Hence sg = (time, room, day, lecturer, exam), i.e. each gene is represented by a five-dimensional vector.

Accordingly, the following result is desired (Fig. 3):

Result 1:	PC-1 / 80	Mo	4 p.m. – 8 p.m.	Lecturer X	Exam Y

Fig. 3. Desired result for one exam:

As there are (for teacher education students) 261 requests all these requests are combined in the structural vector SV, i.e. SV = (sg_1, sg_2, …, sg_{261}); hence each sg is itself a vector. An RGA individual, therefore, for the room allocation problem consists of an 8-dimensional RV, a SV of 261 dimensions, and a vector CV of different connections between RV and SV.

Fig. 4 shows an according RGA-individual:

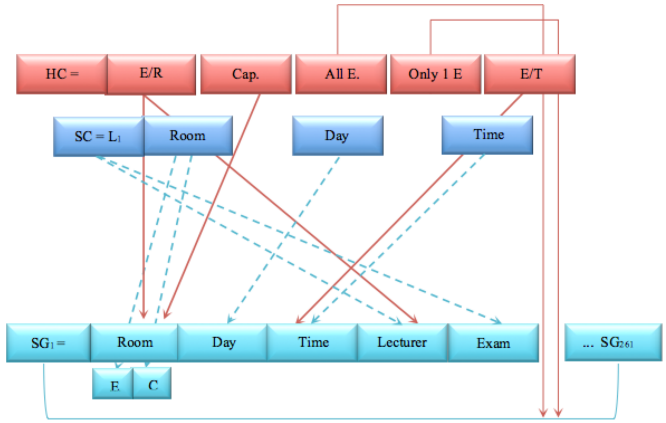

Fig. 4. A RGA individual. HC = Hard Constraints; SC = Soft Constraints for $lecturer_i$; SG_i = Structural Gene ($result_i$); 'All exams' and 'Only 1 exam in a room' are constraints, which check all structural genes.

The constraints, i.e. regulatory genes, are binary coded; mutation, hence, means simply switching them on and off. If a constraint is switched off it plays no role for the structural genes connected with this regulatory gene; accordingly the constraint is not taken into account by the fitness evaluating of these structural genes (see below). However, the results given below refer to a version where the regulatory genes are not varied (see above) and are all switched on. It is possible to use other versions where some constraints should not be taken into regard and are accordingly switched off. The connections are in a figurative sense weighted: The weight values describe the probabilities of the variation of those structural genes that are captured by the specific connections. Mutation of the connections means the exchange of connections.

Variations of the structural genes mean basically the exchanges of one component of a gene by a component from another gene, for example the exchange of the exam (mutation). Crossover in a varied form of "one-point-crossover" of course means the exchange of parts of the whole structural vector with parts from another individual, avoiding wrong duplications.

Obviously a RGA individual is optimal if it fulfills all conditions for all requests, i.e. if no constraints are violated. The task of the RGA in this case is accordingly to minimize the number of violated constraints as far as possible, i.e. to search for a minimum. The fitness function takes this into regard by steering the system to fitness values as low as possible. Only the structural genes are of course evaluated. A RGA individual hence gets its fitness value by a) computing the number of violated constraints for each gene, multiplied by the punishment values for each constraint, and b) by summing the fitness values of the genes of the whole individual:

$$fit(ind) = \sum_i (vc_i * pv_i) \quad (3)$$

vc_i is the number of violated constraints and pv_i the punishment values of the different constraints.

Factually the model is a bit more complicated than shown in this description but for a basic understanding of the model it is sufficient to show the general principles.

Fig. 5 shows the RGA model for this problem:

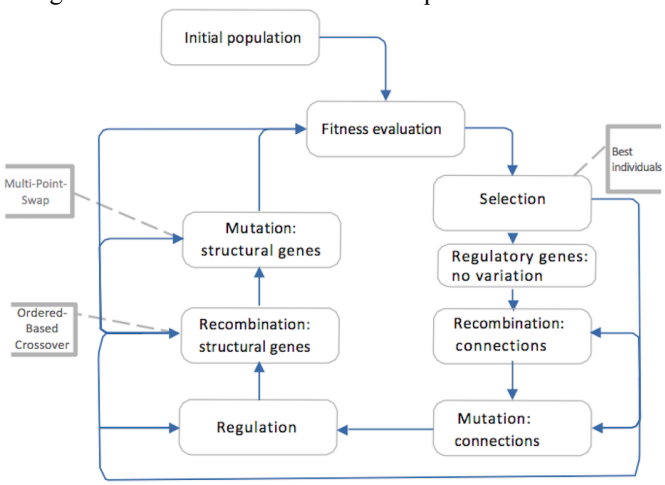

Fig. 5. Schema of the RGA model for the room allocation problem

V. Experiments and Results

The hard constraints, as mentioned, all have the punishment values of 1.0. The soft constraints are valued with 0.9 for wished day, 0.5 for wished room (with the exception of PC- and lab-rooms with a value of 1.0), and 0.8 for wished time. To be sure, in particular the values for the soft constraints are an additional parameter that could – and perhaps should – be changed, for example the value for "wished room". This condition is very difficult to fulfill, especially for the examination tabling.

Because the variation of the regulatory genes makes with regard to content not much sense for this problem we decided to omit this vector from the variation operations. According to the general variation schema VS described in section 2 we chose the option VS = ((0,0), (1,1), (1,1)), i.e. mutation and crossover were applied to both the connection vector and the

structural vector. According to our experiences we chose rather small values of the genetic operators, as shown in Table II:

TABLE II. PARAMETER VALUES FOR THE RGA

Parameters	Parameter-Values
Mutation structural vector (SV)	0,01
Crossover structural vector (SV)	0,5
Mutation vector of connections (VC)	0,1
Crossover vector of connections (VC)	0,1
Mutation regulator genes (RG)	0,0
Crossover regulator genes (RG)	0,0
Population size	200
Iterations	250

We used an elitist version by taking over the best two individuals to the next generation.

For the experiments each time a population of 200 individuals was generated at random. According to preliminary experiments 250 iterations were always enough because the RGA obtained no significant improvements after 250 steps. Figs. 6 - 9 show the best-obtained result, namely the optimum of 0.0, and the possibility to organize all exams within a week; it can be noted that the RGA has solved the task in an unexpected short time:

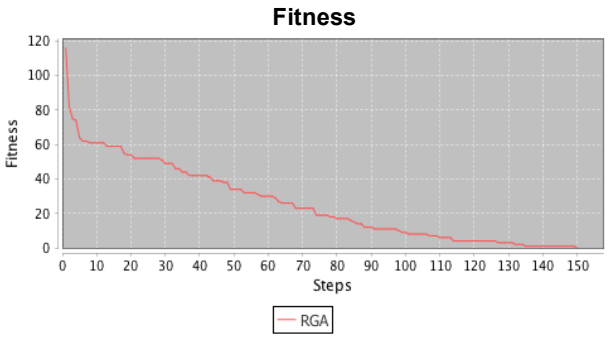

Fig. 6. Fitnesstrend of the RGA. After 150 steps the system reaches the optimum of 0.0.

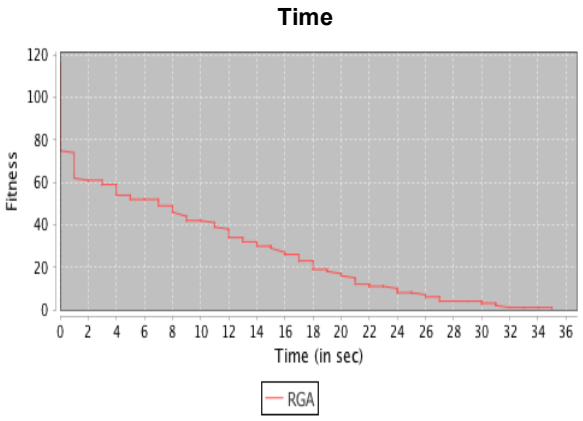

Fig. 7. Needed time for the result

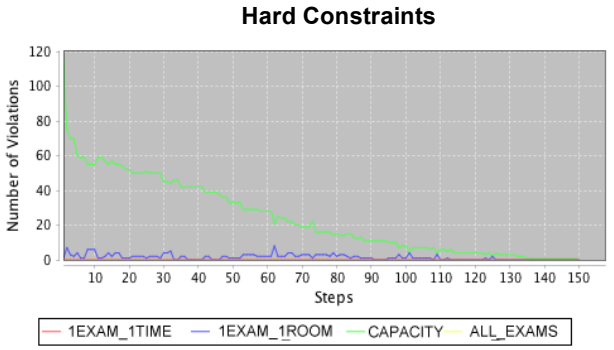

Fig. 8. Development of the hard constraints

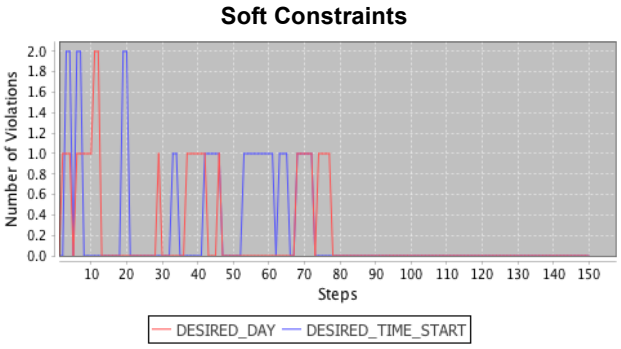

Fig. 9. Development of the soft constraints

Fig. 8 shows in particular that the most difficult problem for the RGA was the constraint of the capacity of the necessary rooms. Fig. 9 demonstrates that the solutions of the soft constraints varied rather much before the RGA found a solution for these constraints too.

Comparison with a Genetic Algorithm

For a comparison of the RGA with a Genetic Algorithm (GA) we have introduced in the GA a "repair mechanism" to make sure that the constraints are also taken into account and that each individual is a correct one. In the first experimental comparison we have used the same parameters for the GA as for the RGA in the previous experiments, namely a population of 200 individuals, 250 iterations, 0,01 for mutation and 0,5 for crossover (cf. Table I). The GA was not able to perform the task in a satisfactory way. Fig. 10 and 11 show the results:

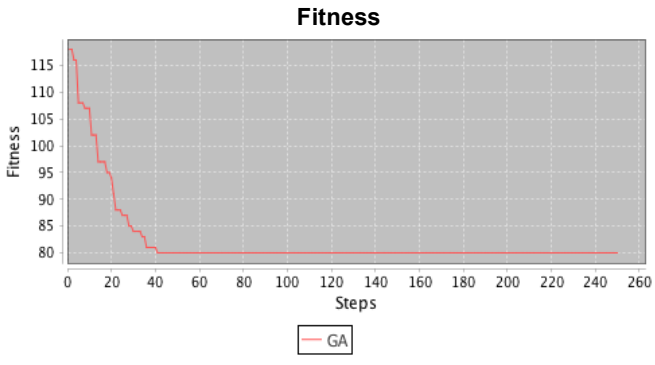

Fig. 10. Development of fitness values

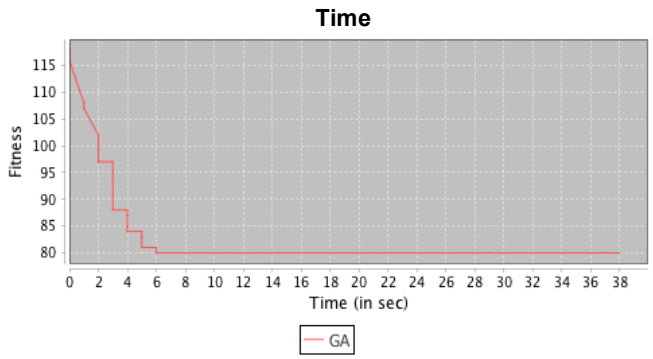

Fig. 11. Needed time for the results

The optimization results show that the fitness value of the GA remains near to 80 (Fig. 10), and that it reaches this local optimum after 6 seconds (Fig. 11).

In the next experiments we choose the most favorable parameters according to our experiments for the GA and the same values for the RGA, shown in Table III:

TABLE III. PARAMETERS FOR THE GA

Parameters	Parameter-Values
Mutation	0,5
Crossover	0,9
Population size	500
Iterations	800

The results are shown in the Fig. 12 and 13:

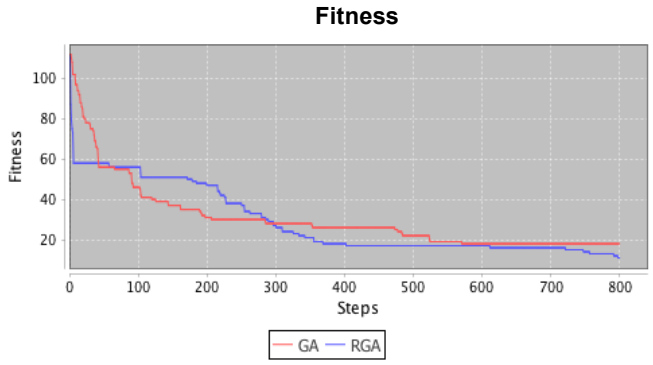

Fig. 12. Development of the fitness trend GA and RGA in comparison

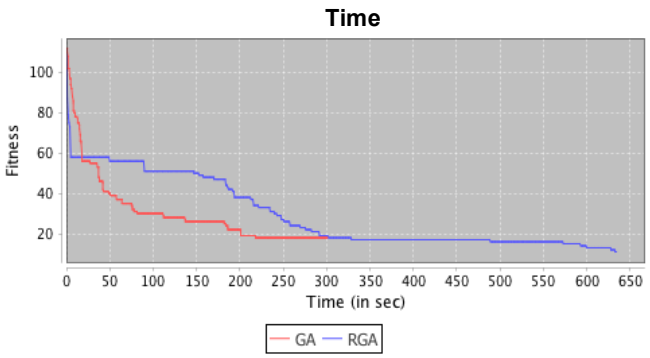

Fig. 13. Development of the needed time GA and RGA in comparison

One sees that the GA reaches a fitness value of 18, which is a very good result, and the RGA reaches a fitness value of 12. The time needed for the results is again different, as the GA needs only 300 seconds, the RGA about 640. This result is typical for all experiments. The GA never reaches the fitness values of the RGA, but is faster in the processing time, because of the needed number of iteration.

Taking into account that the RGA does not need a population larger than 200 and only 150 iterations to get the global optimum, the RGA obviously performs significantly better. Apparently the GA and the RGA need different parameter values to achieve their respective best results. This is also the case with the necessary and suited size of the populations; the GA needed a much larger population than the RGA. Best values for the RGA with suited parameters show an impressive advantage of the RGA: The RGA reached a global optimum; the GA could never do this but remained in local optima.

We are of course aware that there are many different meta heuristics that could be compared with the RGA, in particular approaches that are similar to the RGA. Yet in this rather short article we had to be content with only one of the best-known techniques, namely a standard GA, as comparison.

Of course the problem of examination timetabling can be solved by other algorithms too (cf. the list of references). Yet the obvious practical success of the RGA is a positive indication that it is worthwhile to investigate the possibilities of the RGA by its application to other complex optimization problems. In addition, we compared the RGA with an according GA with respect to several other problems. In most cases the RGA performed significantly better and in several cases only the RGA found a solution at all [13], as is also the case in this example.

VI. Final Remarks

The positive results of the RGA are in accord with results from previous experiments, namely that perhaps the RGA because of its topological structure performs best in the case of problems that can be structured the same way, i.e. as a two-dimensional topological space [13]. At present this is just a hypothesis but it is temporarily confirmed by the results of a room allocation problem [7] and the examination timetabling. We shall analyze this hypothesis in further experiments and applications.

The RGA described in this article is still a prototype in the sense that several user-friendly improvements are necessary for its usage in practice. The most important one is optimizing the time between the exams and to make sure that the students have not more than one exam per day, which is not the case at present.

An additional enlargement is to vary the RGA by allowing double or more exams in the same room to avoid rented locations: The mentioned factor, e.g. 4, means that the room must have at least four times as many places as there are students. In other words, between two students there should be four empty places. If one organizes four exams of four different disciplines then it would be possible to place students of different disciplines together without empty places between them. If the lecturers, hence, wish to have a factor 4 than four different exams can be written at the same time if the students are placed accordingly – for example a student of chemistry besides one of sociology, this one besides a student of literature, and so on. In this case, if there are 300 students for each exam, only one room with 1,200 places is needed, but for four exams. The RGA has then the task to put together different exams like physics, literature, arts and pedagogics.

The RGA looks on a first sight as a rather complicated system, but it is obviously very efficient and it enables solving the constraint problems in a very elegant and simple way. It will be interesting and worthwhile to investigate its potential further.

References

[1] E. Burke, J.P. Newall, and R.F. Weare, "A memetic algorithm for university exam timetabling", in: G. Goos, J. Hartmanis, J. Leeuwen, E. Burke, and P. Ross, Eds, Practice and Theory of Automated Timetabling, vol. 1153. Berlin, Heidelberg: Springer (Lecture Notes in Computer Science), 1996, pp. 241-250.

[2] E. Burke, J.P. Newall, "A multistage evolutionary algorithm for the timetable problem", in: IEEE Transactions on Evolutionary Computation 3 (1), 1999, pp. 63-74. DOI: 10.1109/4235.752921.

[3] J.A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall, M. Carpio, "Iterated local search using an add and delete hyper-heuristic for university course timetabling", Applied Soft Computing, 40, 2016, pp. 581-593

[4] R. Bellio, S. Ceschia, L. Di Gaspero, A. Schaerf, T. Urli, "Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem", Computers & Oerations Research, 65, 2016, pp. 83-92

[5] H. Babaei, J., Karimpour, A. Hadidi, "A survey of approaches for university course timetabling problem, Computers & Industrial Engineering, 86, 2015, pp. 43-59

[6] J. Lee, S.-P. Ma, L. F. Lai, N. L. Hsueh, and Y.-Y. Fanjiang, University Timetabling through Conceptual Modeling, International Journal of Intelligent Systems, Vol. 20, 2005, pp. 1137-1160

[7] M. Kleine-Boymann, C. Klüver, and J. Klüver, Optimization of Room Allocation Plans at the University Duisburg-Essen with a Regulatory Algorithm, Proceedings of IEEE CEC 2016, in press

[8] M. W. Carter, G. Laporte and S. Y. LEE, Examination Timetabling: Algorithmic Strategies and Applications Journal of the Operational Research Society 47, 1996, pp. 373-383

[9] R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee, A survey of search methodologies and automated system development for examination timetabling, Journal of scheduling, 12, 2009, pp. 55-89

[10] S. Larabi Marie-Sainte, A survey of Particle Swarm Optimization techniques for solving university Examination Timetabling Problem, Artificial Intelligence Review, 44: 537, 2015, DOI: 10.1007/s10462-015-9437-7

[11] M.A.H. Hassan, and O.A.H. Hassan, Constraints Aware and User Friendly Exam Scheduling System, International Arab Journal of Information Technology (IAJIT), Vol. 13, No. 1A, 2016.

[12] W. Erben, "A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling", in: E. Burke and W. Erben, Eds, Practice and theory of automated timetabling. Third international conference, PATAT 2000, Konstanz, Germany, August 16-18, Berlin, New York: Springer (Lecture Notes in Computer Science), 2001, pp. 132-156.

[13] J. Klüver, and C. Klüver, "The Regulatory Algorithm (RGA): A two-dimensional Extension of Evolutionary Algorithms", Soft Computing. Springer, 2015 DOI: 10.1007/s00500-015-1624-6.

[14] J.S. Huxley, "Evolution, The Modern Synthesis", London: Allen and Unwin, 1942.

[15] F. Jacob and J. Monod, "Genetic regulatory mechanisms in the synthesis of proteins". Mol. Bio. 1961, vol. 3, pp. 318-356.

[16] S.B. Carroll, "Endless Forms Most Beautiful. The New Science of Evo Devo and the Making of the Animal Kingdom". London: Weidenfeld and Nicolson, 2006.

[17] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", 1994, Berlin: Springer.

[18] D. Ashlock and W. Ashlock, "Impact of regulatory genes on optimization behavior," Evolutionary Computation (CEC), 2012 IEEE Congress on, Brisbane, QLD, 2012, pp. 1-8. DOI: 10.1109/CEC.2012.6252981

[19] S. Vijayvargiya and P. Shukla, "A Structured Evolutionary Algorithm for Identification of Transcription Factor Binding Sites in Unaligned DNA Sequences". International Journal of Advancements in Technology, 2011 (1) pp. 100-107

[20] S. Cussat-Blanc, K. Harrington, and J. Pollack, Gene Regulatory Network Evolution Through Augmenting Topologies, IEEE Transactions on Evolutionary Computation, Vol. 19, Nr. 6, 2015, 823-837

[21] A. J. Soper, C. Walshaw, and M. Cross, "A Combined Evolutionary Search and Multi Level Optimisation Approach to Graph Partitioning", University of Greenwich, Mathematics Research Report 00/IM/ 58, 2000.

[22] C. Antonio, "A multi level genetic algorithm for optimization of geometrically nonlinear stiffened composite structures", Structural and Multidisciplinary Optimization, 2002, vol. 24 (5), pp. 372-386.