MUTATION EFFECTS IN GENETIC ALGORITHMS WITH OFFSPRING SELECTION APPLIED TO COMBINATORIAL OPTIMIZATION PROBLEMS

Stefan Wagner[(a)], Michael Affenzeller[(b)], Andreas Beham[(c)], Gabriel Kronberger[(d)], Stephan M. Winkler[(e)]

[(a – e)] Upper Austria University of Applied Sciences
School for Informatics, Communications, and Media – Hagenberg
Josef Ressel-Centre *Heureka!* for Heuristic Optimization
Heuristic and Evolutionary Algorithms Laboratory
Softwarepark 11, A-4232 Hagenberg, Austria

[(a)]stefan.wagner@fh-hagenberg.at, [(b)]michael.affenzeller@fh-hagenberg.at, [(c)]andreas.beham@fh-hagenberg.at, [(d)]gabriel.kronberger@fh-hagenberg.at, [(e)]stephan.winkler@fh-hagenberg.at

ABSTRACT
In this paper the authors describe the effects of mutation in genetic algorithms when used together with offspring selection to solve combinatorial optimization problems. In the initial definition of offspring selection stated by Affenzeller et al., offspring selection is applied to each solution after its creation using crossover and optional mutation. Thereby a solution is immediately accepted for the next generation only if it is able to outperform its parental solutions in terms of quality.

It has been shown in several publications by Affenzeller et al. that this additional selection step leads to a better maintenance of high quality alleles and therefore to a better convergence behavior and a superior final solution quality.

Due to the application of offspring selection after crossover and mutation, both operations become directed by the quality of the created solutions. This is in fact a different interpretation of mutation compared to classical genetic algorithms where mutation is used in an undirected way to introduce new genetic information into the search process.

In this contribution the authors propose a new version of offspring selection by applying it after crossover, but before mutation. In a series of experiments the similarities and differences of these two approaches are shown and the interplay between mutation and offspring selection is analyzed.

Keywords: Genetic Algorithms, Combinatorial Optimization, Selection

1. INTRODUCTION

Selection for reproduction represents the main driving force in genetic algorithms that guides the search process through the solution space. By selecting solutions of above average fitness and applying crossover and optionally mutation, genetic algorithms try to combine alleles of high quality in order to obtain better and better solutions. During the search process genetic diversity is usually decreased step by step so that the algorithm is able to converge to solutions of high quality in the end.

However, due to the effects of genetic drift high quality (i.e., relevant) alleles which are required to reach global optimal solutions might be lost in the whole population. This leads to so-called premature convergence which describes a state in which the algorithm is no longer able to create better solutions although it has not reached a global optimum so far (Fogel 1994; Affenzeller 2005). This situation can be compared to the effect of getting stuck in a local optimum in neighborhood-based meta-heuristics.

In general the following three aspects can be identified as the main reasons for the loss of relevant alleles (Affenzeller 2005; Affenzeller, Wagner, and Winkler 2010):

- Some relevant alleles might not be included in the initial population. This especially might be the case if the population size is rather small.
- Relevant alleles might be lost due to the stochastic nature of selection and genetic drift. This is frequently the case in an early phase of the algorithm, when relevant alleles are included in solutions of rather bad quality.
- For many applications of genetic algorithms the applied crossover operators cannot guarantee that the created children are exact combinations of the genetic information of the parents as new alleles might have to be introduced in order to create feasible solutions.

In classical genetic algorithms the only approach to counteract the loss of relevant alleles and therefore to avoid premature convergence is mutation. However, as mutation is used as an undirected operator, the probability to get back relevant alleles by lucky mutations decreases rapidly when trying to solve problems of larger size.

Additionally to mutation, some other ideas are also discussed in literature to reduce the negative effects of premature convergence. Among them the most common representative are preselection (Cavicchio 1970), crowding (De Jong 1975), and fitness-sharing (Goldberg 1989). The main idea of these approaches is to maintain genetic diversity by replacing solutions more frequently which occupy similar regions of the search space (preselection, crowding) or the reduce the fitness value of solutions which are located in densely populated regions of the search space (fitness-sharing). All these three approaches require the definition of a distance measure in order to be able to calculate the similarity of solutions in the search space, and fitness-sharing is additionally quite restricted to fitness proportional selection. As a consequence, these approaches are not applicable in each case. Furthermore, they do not really address the problem of loosing relevant alleles but try to reduce the loss of genetic diversity in general.

In order to develop a more general technique to counteract the loss of relevant alleles and therefore to prolongate premature convergence the authors introduced offspring selection (Affenzeller and Wagner 2005; Affenzeller, Wagner, and Winkler 2010; Affenzeller, Winkler, Wagner, and Beham 2009). The basic idea of this selection model is to consider not only the fitness of the parents when creating new solutions. Additionally, the fitness value of each new child solution created by crossover and optionally mutation is compared with the fitness values of its parents. The child is immediately accepted for the next generation if and only if it outperforms its parents' fitness. This strategy guarantees that the search process is continued mainly with crossover results that were able to mix the properties of their parents in an advantageous way and/or with mutation results that contain relevant alleles. In other words, offspring selection supports *survival of the fittest alleles rather than survival of the fittest chromosomes*. This is a very essential concept concerning the preservation of relevant genetic information stored in the individuals of a population.

2. OFFSPRING SELECTION

In general, offspring selection consists of the following steps (Affenzeller and Wagner 2005):

At first parents are selected for reproduction either randomly or in any other well-known way of genetic algorithms (e.g., fitness proportional selection, linear rank selection, tournament selection). After crossover and optionally mutation have been applied to create a new child solution, another selection step is introduced which considers the success of the applied reproduction procedure. The goal of this second selection step (i.e. the offspring selection step) is to continue the search process mainly with successful offspring which surpass their parents' quality. Therefore, a new parameter called success ratio (*SuccRatio*) is introduced. The success ratio defines the relative amount of members in the next population that have to be generated by successful mating (crossover, mutation).

Additionally, it has to be defined when a solution is considered to be successful: Is a child solution better than its parents, if it surpasses the fitness of the weaker, the better, or some kind of mean value of both? For this purpose a parameter called comparison factor (*CompFactor*) is used to define the success criterion for each created solution as a weighted average of the quality of the worse and better parent (i.e., if the comparison factor is 0, successful solutions at least have to be better than the worse parent, and if it is 1 they have to outperform the better parent).

Based on the comparison factor, the authors decided to introduce a cooling strategy which is similar to simulated annealing. Following the basic principle of simulated annealing, an offspring only has to surpass the fitness value of the worse parent in order to be successful at the beginning of the search process (*CompFactor* is initialized with 0 or a rather small value). While evolution proceeds solutions have to be better than a fitness value continuously increasing between the fitness of the weaker and the better parent (*CompFactor* is increased in each generation until it reaches 1 or a rather high value). As in the case of simulated annealing, this strategy leads to a broader search at the beginning, whereas at the end the search process becomes more and more directed.

After the amount of successful solutions in the next generation has reached the success ratio, the remaining solutions for the next generation (i.e., (1-*SuccRatio*)·|*POP*|) are taken from the pool of solutions which were also created by crossover and mutation but did not necessarily reach the success criterion. The actual selection pressure *ActSelPress* at the end of a single generation is defined by the quotient of individuals that had to be created until the success ratio was reached and the number of individuals in the population:

$$ActSelPress = \frac{|POP| \cdot SuccRatio + |POOL|}{|POP|} \quad (1)$$

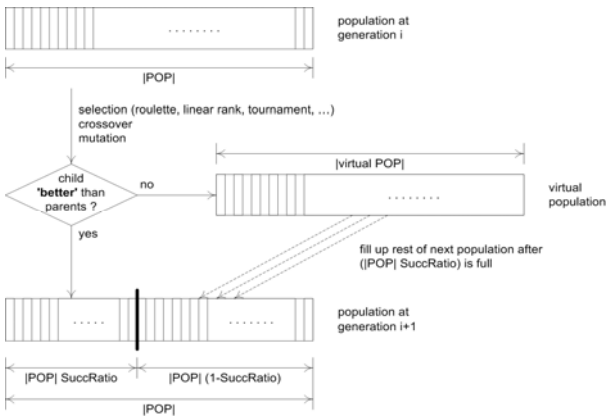

Figure 1: Flowchart of a Classical Genetic Algorithm Extended by Offspring Selection

Figure 1 shows these basic steps of offspring selection and how they are embedded into a classical genetic algorithm.

Furthermore, an upper limit for the selection pressure (*MaxSelPress*) can be defined as another parameter which states the maximum number of children (as a multiple of the population size) that might be created in order to fulfill the success ratio. With this additional parameter offspring selection also provides a precise detector for premature convergence: If the algorithm cannot create a sufficient number of successful solutions ($SuccRatio \cdot |POP|$) even after $MaxSelPress \cdot |POP|$ solutions have been created, premature convergence has occurred and the algorithm can be stopped.

As a basic principle of offspring selection, higher success ratio and comparison factor cause higher selection pressure. Nevertheless, a higher selection pressure does not necessarily cause premature convergence when using offspring selection, as offspring selection also supports the preservation of relevant alleles and not only the preservation of fitter solutions as a whole.

3. OFFSPRING SELECTION AND MUTATION

In its original definition offspring selection is applied to each solution after it has been created by crossover and optionally manipulated by mutation. Thereby offspring selection assures that the search process is continued mainly with solutions which contain a promising combination of the genetic information of their parents (assembled by the crossover operator) and/or which contain high quality alleles that have been added by mutation. This approach leads to a strong direction of the search process. Several experiments have shown that a genetic algorithm with offspring selection is also able to achieve results of high quality, even if selection for reproduction is done randomly and only offspring selection is used to guide the search (Affenzeller 2005; Affenzeller, Wagner, and Winkler 2010).

This is in fact a slightly different interpretation of crossover and mutation compared to classical genetic algorithms. In a classical genetic algorithm the crossover operator is responsible for combining alleles to longer and longer building blocks and mutation is used as an undirected manipulation operator whose purpose is to add new alleles to the population in order to keep the search process alive (Holland 1975). When applying offspring selection both operators, crossover and mutation, are always considered in combination and are directed by the success criterion which has to be fulfilled by the created solutions. Consequently, the mutation operator does not longer serve as an undirected manipulation operator, as it is followed by an additional selection step. This interpretation of mutation is concordant with the way mutation is considered in population genetics and also in evolution strategies (Beyer and Schwefel 2002) where it affects the genotype before and not after selection.

As an alternative, a new version of offspring selection can be easily defined by applying the success criterion after crossover has created a new solution but before the mutation operator is optionally used to manipulate it. In this new version of offspring selection a stronger focus is put on the crossover operator by checking if it was able to combine the genetic information of the parents in a successful way and the mutation operator turns into an undirected operator again. This interpretation of the roles of crossover and mutation is more similar to the classical view on genetic algorithms in which their search process is considered as hyperplane sampling (Whitley 1994).

4. EXPERIMENTAL RESULTS AND ANALYSIS

In order to evaluate and compare the two versions of offspring selection described in the previous sections and to gain a deeper insight into the interplay of offspring selection and mutation, the authors carried out a series of test runs with the ch130 instance of the Traveling Salesman Problem (TSP) taken from the TSPLIB (Reinelt 1991). For all tests HeuristicLab 3.3 (Wagner 2009) was used which provides both versions of offspring selection and can be downloaded from the HeuristicLab homepage at http://dev.heuristiclab.com.

Table 1: Parameter Settings

Parameter	Value
Population Size	500
Parent Selection	Random
Crossover Operators	OX ERX MPX OX, ERX and MPX
Mutation Operators	2-opt 3-opt 2-opt and 3-opt
Mutation Probabilities	1% 5% 10% 20%
Elites	1
Offspring Selection	Before Mutation After Mutation
Success Ratio (*SuccRatio*)	1.0
Comparison Factor (*CompFactor*)	1.0
Maximum Generations	1000
Maximum Selection Pressure (*MaxSelPress*)	250

In Table 1 the algorithm's parameter settings are shown which have been used for the tests. In order to highlight the effects of offspring selection before and after mutation, random parent selection and a success ratio and a comparison factor of 1.0 have been applied. Furthermore, several typical crossover and mutation operators for solving the TSP (Larranaga 1999) have been used in combination with different mutation

probabilities. The crossover and mutation operators have also been applied in combination which means that each time a crossover or mutation operator had to be applied, it was chosen randomly.

Table 2: Relative Difference to the Optimal Solution (Offspring Selection after Mutation)

Mut. Op.	Mut. Prob.	Mean	Standard Deviation
2-opt	1%	0,1359	0,1160
	5%	0,0840	0,0726
	10%	0,1052	0,1005
	20%	0,1038	0,1020
3-opt	1%	0,1288	0,1150
	5%	0,1388	0,1201
	10%	0,1410	0,1262
	20%	0,1455	0,1140
2-opt and 3-opt	1%	0,1400	0,1263
	5%	0,0829	0,0851
	10%	0,1226	0,1152
	20%	**0,0671**	0,0744

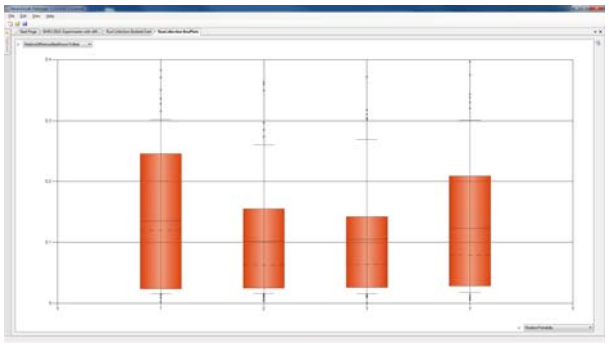

Figure 2: Relative Difference to the Optimal Solution for 1%, 5%, 10% and 20% Mutation Probability (Offspring Selection after Mutation)

Table 3: Relative Difference to the Optimal Solution (Offspring Selection before Mutation)

Mut. Op.	Mut. Prob.	Mean	Standard Deviation
2-opt	1%	0,1307	0,1212
	5%	0,0933	0,0701
	10%	0,0409	0,0325
	20%	0,0222	0,0103
3-opt	1%	0,1525	0,1258
	5%	0,1054	0,1002
	10%	0,0540	0,0684
	20%	0,0240	0,0139
2-opt and 3-opt	1%	0,1156	0,1017
	5%	0,0928	0,0836
	10%	0,0683	0,0799
	20%	**0,0190**	0,0099

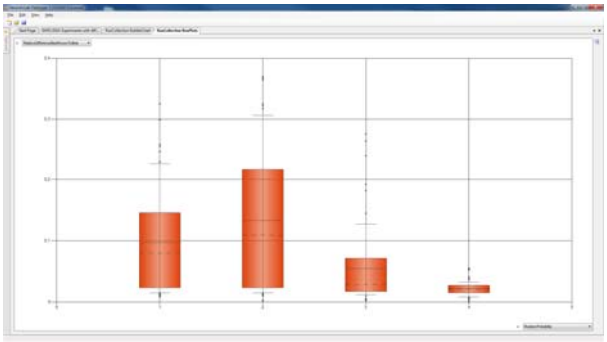

Figure 3: Relative Difference to the Optimal Solution for 1%, 5%, 10% and 20% Mutation Probability (Offspring Selection before Mutation)

For each parameter configuration, 5 independent runs have been executed which gives a total sum of 480 runs. In Table 2 and Figure 2 the relative difference of the best found solution to the global optimal solution is analyzed for the case when offspring selection is done after mutation (classical offspring selection); Table 3 and Figure 3 show the same results for applying offspring selection before mutation (new version).

Furthermore, also the number of evaluated solutions is analyzed for both cases in Table 4, Figure 4, Table 5 and Figure 5.

Table 4: Evaluated Solutions (Offspring Selection after Mutation)

Mut. Op.	Mut. Prob.	Mean	Standard Deviation
2-opt	1%	1.567.880,00	429.171,66
	5%	1.722.365,00	380.228,61
	10%	1.705.435,00	387.580,38
	20%	1.809.925,00	415.801,80
3-opt	1%	1.648.370,00	476.494,38
	5%	1.643.395,00	442.052,44
	10%	1.693.385,00	462.979,53
	20%	1.686.315,00	426.681,99
2-opt and 3-opt	1%	1.633.605,00	515.648,60
	5%	1.773.865,00	436.819,40
	10%	1.621.055,00	393.168,40
	20%	1.929.205,00	480.513,97

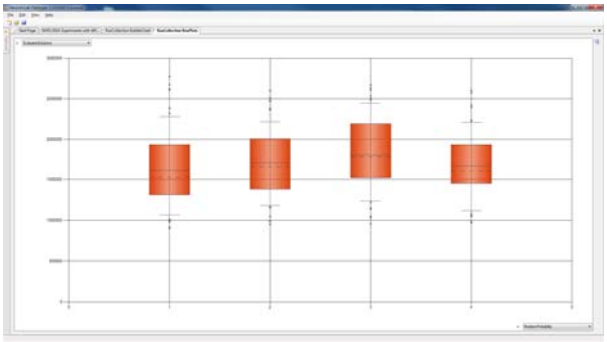

Figure 4: Evaluated Solutions for 1%, 5%, 10% and 20% Mutation Probability (Offspring Selection after Mutation)

Table 5: Evaluated Solutions
(Offspring Selection before Mutation)

Mut. Op.	Mut. Prob.	Mean	Standard Deviation
2-opt	1%	1.637.656,75	405.592,54
	5%	1.857.213,60	382.548,56
	10%	4.957.905,00	4.498.932,66
	20%	31.470.516,65	6.134.381,08
3-opt	1%	1.624.311,45	464.574,57
	5%	1.997.821,55	524.652,97
	10%	18.435.419,90	17.615.456,28
	20%	24.561.314,80	4.869.505,18
2-opt and 3-opt	1%	1.697.508,60	416.604,78
	5%	1.883.439,00	444.074,17
	10%	6.812.471,10	10.224.809,35
	20%	27.093.109,30	3.656.663,69

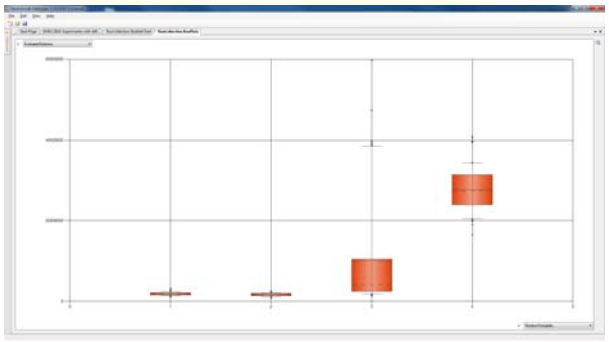

Figure 5: Evaluated Solutions for 1%, 5%, 10% and 20% Mutation Probability
(Offspring Selection before Mutation)

Obviously the algorithm's behavior changes if offspring selection is applied before mutation, especially when working with high mutation probabilities. In the classical version of offspring selection (offspring selection after mutation) changing the mutation rate does not make much difference. The relative difference to the optimal solution as well as the number of evaluated solutions do not change notably (see Table 2, Figure 2, Table 4, Figure 4). However, when applying offspring selection before mutation, the mean value and the standard deviation of the relative difference to the optimal solution decrease significantly and the mean value and the standard deviation of the number of evaluated solutions increase significantly when increasing the mutation probability (see Table 3, Figure 3, Table 5, Figure 5).

These results show that applying offspring selection before mutation and using rather high mutation probabilities leads to a longer execution time, more solution evaluations and more robust algorithms. In fact, this is quite reasonable and consistent with the theory of genetic algorithms. In the classical version of offspring selection the effects of mutation are dominated by the offspring selection step. Therefore, mutation is not undirected anymore and its characteristics as diversification method are lost.

In the new version of offspring selection (offspring selection before mutation) mutation is more used in a way similar to standard genetic algorithms as the result of a mutation is not going through an additional selection process anymore. By this means, mutation becomes a diversification method again which leads to a longer run time, a broader search in the solution space, significantly more solution evaluations, but also more robust results.

5. CONCLUSIONS

In this contribution the authors focused on offspring selection which has been proposed by Affenzeller et al. to counteract the loss of relevant alleles and to prolongate premature convergence. A new version of offspring selection was presented in which the offspring selection step is not applied after mutation but before mutation.

In a series of tests the authors analyzed the similarities and differences of these two versions of offspring selection when solving the Traveling Salesman Problem.

It was shown for the original version of offspring selection (offspring selection after mutation) that the effects of mutation are dominated by the offspring selection step and that mutation therefore does not act as a diversification method anymore. However, when applying offspring selection before mutation the algorithm's behavior is significantly different, especially when using high mutation probabilities. As in this case the result of a mutation is not going through an additional selection process anymore, mutation is applied in an undirected way again and therefore leads to a diversification of the search process. By this means, the algorithm evaluates more solutions and becomes more robust when applying offspring selection before and not after mutation.

In the future the authors are going to focus on a comparison of the robustness of both offspring selection versions when evaluating approximately the same number of solutions. Furthermore, additional test series with other TSP instances and also other combinatorial optimization problems should be done to enable an even more reliable analysis and interpretation.

ACKNOWLEDGMENTS

The work described in this paper was done within the Josef Ressel-Centre *Heureka!* for Heuristic Optimization (http://heureka.heuristiclab.com) sponsored by the Austrian Research Promotion Agency (FFG).

REFERENCES

Affenzeller, M. and Wagner, S., 2005. Offspring selection: A new self-adaptive selection scheme for genetic algorithms. *Proceedings of ICANNGA 2005*, pp. 218–221. 21st–23rd March 2005, Coimbra, Portugal.

Affenzeller, M., 2005. *Population genetics and evolutionary computation: Theoretical and practical aspects*. Linz: Trauner Verlag.

Affenzeller, M., Wagner, S. and Winkler, S.M., 2010. Effective allele preservation by offspring selection: An empirical study for the TSP. *International Journal of Simulation and Process Modelling*, 6 (1), 29–39.

Affenzeller, M., Winkler, S.M., Wagner, S. and Beham, A., 2009. *Genetic algorithms and genetic programming – Modern concepts and practical applications*. Boca Raton: CRC Press.

Beyer, H.G. and Schwefel, H.P., 2002. Evolution strategies: A Comprehensive Introduction. *Natural Computing*, 1 (1), 3–52.

Cavicchio, D.J., 1970. *Adaptive search using simulated evolution*. Thesis (PhD). University of Michigan.

De Jong, K., 1975. *An analysis of the behavior of a class of genetic adaptive systems*. Thesis (PhD). University of Michigan.

Fogel, D., 1994. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5 (1), 3–14.

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

Holland, J.H., 1975. *Adaption in Natural and Artificial Systems*, University of Michigan Press, Ann Harbor.

Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I. and Dizdarevic, D., 1999. Genetic algorithms for the traveling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13, 129–170.

Reinelt, G., 1991. *TSPLIB – A traveling salesman problem library*. ORSA Journal on Computing, 3, 376–384.

Wagner, S., 2009. *Heuristic optimization software systems – Modeling of heuristic optimization algorithms in the HeuristicLab software environment*. Thesis (PhD). Johannes Kepler University Linz.

Whitley, D., 1994. *A Genetic Algorithm Tutorial*. Statistics and Computing, 4, 65–85.

AUTHORS BIOGRAPHIES

STEFAN WAGNER received his MSc in computer science in 2004 and his PhD in engineering sciences in 2009, both from Johannes Kepler University (JKU) Linz, Austria; he is professor at the Upper Austria University of Applied Sciences (Campus Hagenberg). Dr. Wagner's research interests include evolutionary computation and heuristic optimization, theory and application of genetic algorithms, model driven software development and software engineering.

MICHAEL AFFENZELLER has published several papers, journal articles and books dealing with theoretical and practical aspects of evolutionary computation, genetic algorithms, and meta-heuristics in general. In 2001 he received his PhD in engineering sciences and in 2004 he received his habilitation in applied systems engineering, both from the Johannes Kepler University of Linz, Austria. Michael Affenzeller is professor at the Upper Austria University of Applied Sciences, Campus Hagenberg, and head of the Josef Ressel Center *Heureka!* at Hagenberg.

ANDREAS BEHAM received his MSc in computer science in 2007 from Johannes Kepler University (JKU) Linz, Austria. His research interests include heuristic optimization methods and simulation-based as well as combinatorial optimization. Currently he is a research associate at the Research Center Hagenberg of the Upper Austria University of Applied Sciences (Campus Hagenberg).

GABRIEL K. KRONBERGER received his MSc. in computer science in 2005 from Johannes Kepler University Linz, Austria. His research interests include parallel evolutionary algorithms, genetic programming, machine learning and data mining. Currently he is a research associate at the Research Center Hagenberg of the Upper Austria University of Applied Sciences.

STEPHAN M. WINKLER received his MSc in computer science in 2004 and his PhD in engineering sciences in 2008, both from Johannes Kepler University (JKU) Linz, Austria. His research interests include genetic programming, nonlinear model identification and machine learning. Since 2009, Dr. Winkler is professor at the Department for Medical and Bioinformatics at the Upper Austria University of Applied Sciences, Campus Hagenberg.