Evolving Solutions for Design and Management Tasks on Computers

H.–P. Schwefel

University of Dortmund, Department of Computer Science
Chair of Systems Analysis, D-44221 Dortmund, Germany
E-mail: schwefel@LS11.cs.uni-dortmund.de
URL: http://LS11-www.cs.uni-dortmund.de

ABSTRACT

Uninitiated may find it strange that *artificial evolution* resides among a class of problem solving methods belonging to a field named *computational intelligence*. Some people still believe that nature's trial-and-error way to adapt subsystems to their environment is a prodigal game at dice that has led to admirable results only due to vast resources of time and space. A rather simple *gedankenexperiment*, however, reveals that all 10^{80} most elementary particles in the universe together with 10^{60} tiniest time steps since the begin of time cannot explain the development of even simplest bacterial genomes by pure random sampling. Organic evolution must have found a more efficient way to develop clever individuals and manage complex systems. Since about forty years now, a couple of scientists have tried to mimic this process, and they have learned to exploit some *tricks of life* for solving an amazing variety of design and management tasks. This paper tries to give an overview of some recent applications as well as a summary of what we know about the general behavior of evolutionary algorithms.

1 COMPUTATIONAL INTELLIGENCE

The term *Computational Intelligence* (CI), as coined by Bezdek in [1], basically subsumes three approaches to make use of natural problem solving procedures:

- Artificial Neural Networks (NN),
- Fuzzy Logic (FL), and
- Evolutionary Algorithms (EA).

Whereas the two former approaches are centered on imitating processes within individual brains, the latter makes use of the collective behavior of populations. The common characteristics are thought to be:

- Adaptivity,
- Fault tolerance,
- Inherent parallelism,
- Optimal balance between processing speed and accuracy.

In contrast to classical *Artificial Intelligence* (AI) methods, CI incorporates subsymbolic knowledge aggregation and information processing. Another feature of CI techniques is their mutual complementarity, which has already led to a couple of hybrid systems. In the following, the focus will be set on EA, mainly.

2 EVOLUTIONARY COMPUTATION

Since long ago, attempts have been made to model evolutionary processes. Among the time-continuous models the Fisher-Eigen approach [see 2] may be the best known one. Time-discrete models became popular with the first digital computers on campuses. Among them are three, that were devised independently of each other at different places and are still in use in a more or less modified manner:

- Evolutionary Programming (EP),
- Genetic Algorithms (GA), and
- Evolution Strategies (ES).

Genetic Programming (GP) is a more recent offspring of GA, especially devised to evolve computer programs. A history of those early days of evolutionary algorithms may be found in the *Handbook of Evolutionary Computation* (EC) [3], an annotated collection of early papers in the field in D. Fogel's *Evolutionary Computation – The Fossil Record* [4]. Different motives are at work behind mimicking evolutionary mechanisms. One of them has been the exploration of life's mechanisms for a better understanding of natural phenomena, an other the exploitation of the ability of such processes to solve adaptation and optimization problems even if traditional methods fail.

Though the basic ideas of EC were originated in the late Fifties and early Sixties of this century, broad acceptance and use for practical applications cannot be reported before the mid-Eighties. According to Alander's [5] amazing annotated bibliography, several hundreds of articles are published now annually, and there is hardly any field of engineering and management, where the algorithms have not yet been applied.

For a collection of management and business examples see Biethahn and Nissen [6]; other examples are more or less scattered in the proceedings of the major conference series:

- International Conference on Genetic Algorithms (ICGA, since 1985, biennial),
- Foundations of Genetic Algorithms (FOGA, since 1990, biennial),
- Parallel Problem Solving from Nature (PPSN, since 1990, biennial),
- Annual Conference on Evolutionary Programming (EP, since 1992),
- IEEE International Conference on Evolutionary Computation (ICEC, since 1994, annually),

- Genetic Programming Conference (GP, since 1996, annually).

and in the journals of the field:

- Evolutionary Computation (The MIT Press, since 1993),
- IEEE Transactions on Evolutionary Computation (since 1997),
- Evolutionary Optimization (E-journal, since 1998),
- Genetic Programming and Evolvable Hardware (Kluwer Academic Press, forthcoming).

There are at least twenty other regularly scheduled conferences embracing the topic EA/EC as well as more and more journals editing special issues on various aspects from that scope.

3 MAIN INGREDIENTS OF EVOLUTIONARY ALGORITHMS

This is not the place to give a formal description of any of the many incarnations of EA. The reader may get such information from numerous books, especially the Handbook of EC [7], already mentioned above. Brief overviews may be found in [8,9,10]. However, the main ingredients of all basic algorithms will need some comments in order to refer to them in the following sections. Though not all EA have been devised for serving to solve optimization tasks, this very common kind of application will be assumed here to explain algorithmic features.

Population

In contrast to classical optimization procedures, EA operate with several instead of just one search points in the space of the decision variables. Each search point is called *individual* and the set of those existing at a given time *population*. The objective function to be maximized or minimized is called *fitness* function, though this is not correct in biological terms.

Representation

Canonical GA use a binary representation of the variables characterizing an individual. Except for Boolean functions describing the relationship between the binary *genotype* and the individual's fitness, a mapping of the bitstrings onto the set of decision variables (the *phenotype*) is required. So-called Hamming cliffs exist, even if the Gray code is used, if more than one bit is flipped at a time within the bits encoding one variable.

The earliest ES operated with equidistant discrete (integer) variables describing experimentally tested devices like slender bodies in a flow. Despite of that, ES are better known in conjunction with real-coded variables, since most of the theoretical results have been derived for this domain. Counterexamples are Rechenberg's [11] butterfly mimicry simulations, Rudolph's [12] investigations of mutation processes for integer variables, computer experiments with binary variables and multicellular individuals [13], and even variable-length *genomes* already used in 1968 [14]. Likewise early, the individuals of ES were represented as a set of two vectors each, one vector for the object variables, the other for the most important *internal strategy parameters*. In the simplest case, just one common standard deviation, encoding the *mean step size* of all Gaussian mutations, was used. For a more sophisticated version, see [15].

Originally, EP used finite state automata as individuals, but recently ES-like representations have become common practice. In GP, one typical representation is a tree of S-expressions, and variable length is a must. From the application point of view, more often than not, mixed representations are necessary. Some examples will be given below.

Variation

In nature we distinguish between *mutation* and *recombination*, the latter kind of variation being connected with sexual inheritance. EP does not make use of recombination, because species are thought of as evolving units. GA (and GP) emphasize *crossover* of two genomes as main or only *genetic* operator, used with a probability of about 60%. Bit flipping as mutation operator has been introduced mainly to avoid the premature loss of diversity at single positions of the evolving unit, the genome. A typical mutation rate is 0.1%. In general, both probabilities are fixed over one run of a GA. EP mutations will not be handled explicitly, because they were of very special type for varying finite state automata long ago, and they are very similar to ES mutations now.

Mutation and recombination, both at 100% rates, are common practice in ES - except for the earliest version with one parent and one descendant, used in experimental optimization without computer. Since theoretical results for mutations have been found earlier (and easier), the false impression has been circulated that recombination is considered less important within ES. The contrary is true. Recombination, especially intermediary recombination of the internal strategy parameters, is essential for achieving high convergence rates as well as auto-adaptation of the strategy parameters (like variances and covariances used for controlling the mutation probabilities or probability densities) [16]. Correlated variations may be thought of as phenotypic changes caused by the corresponding set of parameters that form the genotype by a mapping process in the *epigenetic apparatus*. The evolving unit, thus, is the individual or cell (most EA individuals are not yet multicellular), a point of view shared by many biologists.

Generation transition and selection

Most EA incarnations are using a rather simple scheme for the generation transition. They start with a set of let us say μ parents, apply their variation operator(s) as often as necessary to produce λ offspring, evaluate all individuals according to the given fitness (objective) function, and apply a selection operator to decide upon which genomes, individuals, or species will become parents during the next generation. A closer look, however, reveals some interesting differences in detail. One of them concerns the number of offspring per parent within a generation, another the maximum life span of individuals, and a third the mating frequency.

Only ES work with a birth surplus, i.e. $\lambda > \mu$, all other EA not. The number of descendants may even go down to $\mu = 1$ for all EA including ES, but only in case of *elitist* selection, which means that parents *survive* if there are not enough offspring that are at least equal in terms of their fitness values. Thus, elitist selection in principle allows infinite life span for individuals.

A modern EP version uses a set of tournaments among a random subset of all old and new species, and those species that gain highest scores will enter the next generation. Canonical GA are based on proportional non-elitist selection, a form that is out of practice, but still plays a role in theory. All offspring are given a chance to mate (different partners, in general), but this chance is proportional to their share of the sum of all fitness values within the current population. An alternative exists by just ranking the offspring, but tournament selection has become the most frequently used form of selection within GA, nowadays.

Contemporary ES [15] use a life span delimiter within the genome, so that the older non-elitist (μ, λ) version forms one extreme with maximum life span of one generation, and the older elitist $(\mu + \lambda)$ version resembles the other extreme with infinite maximum(!) life span. The life span is shorter, of course, if better offspring appear. Among all offspring plus those parents whose life span has not yet expired, a truncation selection takes place, so that only the μ best individuals reach the next reproduction cycle. The mating probability does not depend on fitness or age.

Especially the selection operators as mentioned above model quite different natural phenomena. One may resume that

- proportional and ranking selection model what has been called sexual choice;
- tournament selection models competition for survival;
- truncation selection models environmental testing.

A completely different, non-generational, scheme of an EA has been devised recently for multi-criteria as well as dynamic optimization. It is based on a predator-prey model and spatially distributed individuals, and moreover, it no longer needs any synchronization of birth and death processes [17]. The reader must be referred to the literature for further details.

4 ONE OLD AND SOME RECENT APPLICATIONS

Just to demonstrate the versatility of an evolutionary approach to design and management challenges given from real-world applications, a few examples will be mentioned, referenced, and briefly commented subsequently:

- The experimental design of a Laval-nozzle for a one-component two-phase supersonic flashing flow;
- The management of nuclear reactor refueling processes;
- The design of chemical processing plants;
- The management of energy supply systems.

Nozzle length and shape optimization
Long ago, when three-dimensional supersonic flows with turbulent wall friction and non-equilibrium phase transitions could not yet be simulated on a computer, the question was posed, how to shape the contour of a flashing nozzle in order to maximize the efficiency of transforming the thermal energy at a nozzle entry into kinetic energy at its exit. Such task was solved with a simple $(1 + 1)$ ES using binomially distributed mutations and decreasing variance during two weeks of experimenting with hot water at the power station of the Technical University of Berlin [14].

The nozzle was formed by conically bored brass segments of 1 cm width that could be assembled to form an inner 3D contour with diameters varying from 6 mm at the throat to a maximum of 36 mm in steps of 2 mm. With no more than 300 of such segments it would have been possible to set up 10^{30} different nozzles with same diameters at adjacent ends of the rings. Whether or not the globally optimal shape was finally found, the result was better than expected and far better than any nozzle contour known before. Just 300 experiments had been sufficient to do the job. However, some more time and effort was necessary to understand why the result was so good despite of the rather strange contour found (see http://LS11-www.cs.uni-dortmund.de/people/kursawe/Demos/Duese/duese GIFE.html).

Since the proper length of the nozzle was not known in advance, two additional variables were added to the list of the contour diameters: the numbers of segments between entrance and throat and between throat and nozzle end. By imitating gene duplication and gene deletion, both the convergent as well as the divergent parts became variable in length, the list of diameters thus shorter or or longer. Without this trick no really good result could have been achieved, because the best nozzle turned out to be much longer than expected.

Fuel rod management for nuclear reactors
About once annually, the fuel assemblies of a nuclear pressurized water reactor have to be rearranged. Some of them are replaced by new ones, others have to be carried into new positions and/or just to be given a new orientation. This operation includes the change of fuel rods between the core and a stock of used but still usable assemblies. The corresponding optimization problem is a combinatorial one, and the computing time to simulate one year of operation are substantial. On the one hand, experts already have got a lot of experience, though no proven theory, how to do their job; on the

other hand, any slight improvement would save a lot of money. In [18] the task is described in more detail. At first, one was happy to find solutions from scratch and automatically by means of a modified EA that were nearly as good as those found by experts. Finally, by incorporating some clever heuristics, based on physical and mathematical analyses, into the variation operators, even better than previously known results could be obtained.

Synthesis of heat exchanger networks
Modern chemical plants comprise dozens to hundreds of different process units with interconnecting streams of fluids at different temperatures. Heat exchangers serve as means to reduce production cost by transferring heat from flows to be cooled to those that need additional heat. Other devices are flow splitters and mixers. A cost optimal design of a complete heat exchanger network is a great challenge for an optimization algorithm due to mixed type variables (real, integer, otherwise discrete) and a variable number of devices and their design variables.

Specialized simulation tools are available today, but they do not support the iterative amelioration of parameters automatically. This task has to be done by the design engineer, and rather often a handful of simulation experiments with selection of the best one is called optimization. Devising an EA for solving such kind of problems has been done recently [19].

Most important in this case has been the choice of the proper representation and the design of corresponding genetic operators. Only a graph representation turned out to be manageable for larger arrays of heat exchangers, heaters, and coolers, because a matrix representation, the first attempt, exploded quadratically with the number of both hot and cold flows. The simulation alone already being quite time consuming, the EA was devised to make use of a cluster of workstations in parallel. A (24+1) ES outperformed all other variants with respect to the convergence velocity, but a (128,768) linear-neighborhood ES [20] was more reliable while still preserving reasonable time efficiency.

Power station network management
To balance electricity supply and demand within continentally interconnected networks requires anticipative control of a set of electricity producing units. Kiendl [21] since long makes use of fuzzy rule bases to predict the load and to control the power plants. His Fuzzy-ROSA toolkit deals with positive as well as negative rules by means of hyperfuzzification and hyperdefuzzification, the rules not being extracted from the experts via classical knowledge acquisition, but by evolving the rules in a data-driven way, the data being gathered from observing the experts at work. An ES is used to select the best set of fuzzy rules, while the internal parameters of the ES are controlled during the search for an optimum by means of a fuzzy controller. This type of hand-in-hand cooperation of different CI-methods may become more and more typical for successful hybrid systems in the future. Combining NN with EA has been reported several times; one recent example may be found in [22], the goal being to predict the thermodynamic properties of new chemical compounds from data of their constituents only, that is without experiments.

What is missing to a great extent, are theory-based rules for devising specialized evolutionary algorithms. However, some more insight into their behavior has been found during the last years.

5 THEORETICAL RESULTS

Last, not least, a few comments and hints should be given with respect to theoretical results concerning the convergence, efficiency, and reliability of evolutionary algorithms. Here, we mostly rely upon some recent results of the Collaborative Research Centre (Sonderforschungsbereich) 531 for *The Design and Management of Complex Technological Processes and Systems by Means of Computational Intelligence Methods*, sponsored by the *Deutsche Forschungsgemeinschaft* (DFG).

Many GA users still rely upon Holland's schema theorem [23] and the building block hypothesis. This schema theorem, saying that above average bitstring schemata spread exponentially over time within a population under proportional selection and not too high mutation as well as crossover probabilities, has been revised recently, i.e., more than twenty years after its creation, by Menke [24]. The building block hypothesis, saying that crossover lives from putting together parts of the genome that have selection advantages themselves, holds for linear and other separable fitness functions, but not in general. Salomon's investigations [25] demonstrate that using simple GA on quadratic functions may need in the order of n^m fitness samples if m out of n bits must be flipped simultaneously to achieve any further progress.

Beyer's analyses [26] suggest a different understanding of the benefit of recombination (in ES), called *genetic repair*. Investigations of the acceleration by global intermediary as well as discrete recombination hint to a gain factor proportional to the population size (μ) under certain conditions like parallel computation. Using Cauchy probability density distributions instead of Gaussian ones has sometimes been thought of as being beneficial. Rudolph [27] has shown that this is not true. On the contrary, it may lead to a substantial decrease of convergence velocities, if one does not make use of a spherically symmetric n-dimensional distribution.

Generally, convergence rates of EA can be predicted only for simple fitness functions and simple incarnations of the algorithms. One of the best investigated fitness functions always has been the so-called hypersphere model, a simple quadratic form. Bäck [28], Beyer [29], and Rudolph [30] had already pushed forward the front of insight into ES and other EA during the past years, including preliminary investigations of the acceleration by global intermediary as well as discrete recombination [26], self-adaptation of mutation strengths [31] and the influence of noise [32]. More recently, Oyman [33] has found interesting results for other fitness landscapes, so-called ridge functions. The $(1+1)$ ES selection scheme used with otherwise GA-typical fitness landscapes has led to exact results for two complete function classes [34,35], i.e. linear and unimodal Boolean functions. The same authors have also given a first proof of the effort reducing effect of crossover under similar conditions [36]. For a special function, the savings of function evaluations are super-polynomial. The discouraging no-free-lunch theorem of Wolpert and Macready [37] has been replaced by them by a free appetizer theorem [38], thus encouraging the search for special algorithms that perform better than others on certain classes of optimization problems. First theoretical results for the predator-prey approach to find, within one run of an EA, the whole Pareto set of non-dominated solutions of a multiple criteria optimization problem have been presented by Rudolph [39].

For keeping pace with the ongoing research, it might be worthwhile to have an eye on the report series of the above mentioned SFB 531 at the University of Dortmund (see http://sfbCI.informatik.uni-dortmund.de/reiheci.html).

6 CONCLUSION

Nobody should forget the good old linear and non-linear optimization procedures like conjugate gradient and quasi Newton methods. If they work, they cannot be beaten by evolutionary algorithms. The latter prove to be helpful in many cases where the former fail. But they cannot solve NP hard or NP complete problems in polynomial time. The *curse of dimensions* is valid for them, as well. Under simplest conditions their average rate of convergence is inversely proportionally to the number n of degrees of freedom of the optimization problem. Distances in space growing at least with the square root of n and the effort to evaluate objective functions at least proportional to n, nobody should wonder, that the time complexity often increases cubically, if not worse.

But, EA can make better use of parallel processing environments, may they be parallel computers or clusters of workstations - due to their inherent parallelism. With proper hard- and software tools, linear speedup can be maintained for a certain range of processors involved in the search.

What remains to be said here, is that also EA underlie the tradeoff between robustness and convergence velocity. If one implies incarnations that make them too greedy, they may lose the ability to approximate better solutions or even diverge (e.g. non-elitist versions that are otherwise better suited for the self-adaptation of internal parameters). On the other hand, EA concepts present a rather flexible frame that can quickly be adapted to different situations by a skillful user. Their attractivity stems from the fact that this often is less time consuming than looking for a specialized algorithm, which could reduce the computational effort by an order of magnitude, but would require months to be developed.

REFERENCES

[1] J. M. Zurada et al., Eds., *Computational Intelligence – Imitating Life*, IEEE Press, Piscataway NJ, 1994.

[2] J. Maynard Smith, Models of evolution, *Proc. Royal Soc. London B*, Vol. 219, pp. 315-325, 1983.

[3] K. De Jong, D. B. Fogel, and H.-P. Schwefel, A history of evolutionary computation, In [7], pp. A2.3:1-12.

[4] D. B. Fogel, *Evolutionary Computation – The Fossil Record*, IEEE Press, Piscataway NJ, 1998.

[5] J. T. Alander, *An Indexed Bibliography of Genetic Algorithms, preliminary edition*, J. T. Alander, Espoo, Finland, 1994.

[6] J. Biethahn and V. Nissen, Eds., *Evolutionary Algorithms in Management Applications*, Springer, New York, 1995.

[7] Th. Bäck et al., *Handbook of Evolutionary Computation*, Oxford Univ. Press, New York, and Inst. of Physics Publ., Bristol, 1997.

[8] Th. Bäck and H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, *Evolutionary Computation*, Vol. 1, pp. 1-23, 1993.

[9] Th. Bäck and H.-P. Schwefel, Evolution strategies I: Variants and their computational implementation. In [40], pp. 111-126.

[10] H.-P. Schwefel and Th. Bäck, Evolution Strategies II: Theoretical Aspects, In [40], pp. 127-140.

[11] I. Rechenberg, *Evolutionsstrategie '94*, Frommann-Holzboog, Stuttgart, 1994 (enlarged edition of the PhD thesis of 1971).

[12] G. Rudolph, An evolutionary algorithm for integer programming, In Y. Davidor, H.-P. Schwefel, and R. Männer, Eds., *Parallel Problem Solving from Nature - PPSN III*, Springer, Berlin, 1994, LNCS 866, pp. 139-148.

[13] H.-P. Schwefel and F. Kursawe, On natural life's tricks to survive and evolve, In [41], pp. 1-8.

[14] J. Klockgether and H.-P. Schwefel, Two-phase nozzle and hollow core jet experiments, In D. G. Elliott, Ed., *Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics*, California Inst. of Technology, Pasadena CA, March 1970, pp. 141-148.

[15] H.-P. Schwefel, Parallel problem solving from nature, In A. Kent, J. G. Williams, C. M. Hall, Eds., *Encyclopedia of Computer Science and Technology*, Marcel Dekker, New York, 1997, Vol. 37, Suppl. 22, pp. 225-246.

[16] H.-P. Schwefel, *Evolution and Optimum Seeking*, John Wiley & Sons, New York, 1995 (enlarged translation of the PhD thesis of 1974/75).

[17] M. Laumanns, G. Rudolph, and H.-P. Schwefel, A spatial predator-prey approach to multi-objective optimization, In [42], pp. 241-249.

[18] C. Kappler et al., Refueling of a nuclear power plant: Comparison of a naive and a specialized mutation operator, In [43], pp. 829-838.

[19] B. Groß et al., Optimization of heat exchanger networks by means of evolution strategies, In [43], pp. 1002-1011.

[20] J. Sprave, Linear neighborhood evolution strategy, In A. V. Sebald, L. J. Fogel, Eds., *Proc. 3rd Annual Conf. Evolutionary Programming*, World Scientific, Singapore, 1992, pp. 42-51.

[21] T. Slawinski et al., Data-based generation of fuzzy-rules for classification, prediction and control with the Fuzzy-ROSA method, In *European Control Conf.*, Karlsruhe, Aug./Sept. 1999, VDI/VDE Gesellschaft Mess- und Automatisierungstechnik, (in press).

[22] M. Mandischer, Evolving recurrent neural networks with non-binary encoding, In *Proc. Second IEEE Int'l Conf. Evolutionary Computation*, IEEE Press, Piscataway NJ, Vol. 2, pp. 584-589.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor MI, 1975.

[24] R. Menke, A revision of the schema theorem, Techn. Rep. CI 14/97 of the Collaborative Research Center SFB 531, University of Dortmund, 1997.

[25] R. Salomon, Some comments on evolutionary algorithm theory, *Evolutionary Computation*, Vol. 4, pp. 405-415, 1996.

[26] H.-G. Beyer, Toward a theory of evolution strategies: On the benefit of sex – The $(\mu/\mu, \lambda)$ theory, *Evolutionary Computation*, Vol. 3, pp. 81-111, 1996.

[27] G. Rudolph, Asymptotical convergence rates of simple evolutionary algorithms under factorizing mutation distributions, In J.-K. Hao et al., Eds., *Artificial Evolution: European Conf.*, Springer, Berlin, 1998, LNCS 1363, pp. 275-285.

[28] Th. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[29] H.-G. Beyer, On the 'explorative power' of ES/EP-like algorithms, In V. W. Porto et al., Eds., *Evolutionary Programming VII*, Springer, Berlin, 1998, LNCS 1447, pp. 323-334.

[30] G. Rudolph, *Convergence Properties of Evolutionary Algorithms*, Verlag Dr. Kovač, Hamburg, 1997.

[31] H.-G. Beyer, Toward a theory of evolution strategies: Self-adaptation, *Evolutionary Computation*, Vol. 3, pp. 311-347, 1995.

[32] G. Rudolph, Reflections on bandit problems and selection methods in uncertain environments, In Th. Bäck, Ed., *Genetic Algorithms: Proc. Seventh Int'l Conf.*, Morgan Kaufmann, San Francisco CA, 1997, pp. 166-173.

[33] A. I. Oyman, H.-G. Beyer, and H.-P. Schwefel, Where elitists start limping: Evolution strategies at ridge functions, In [42], pp. 34-43.

[34] S. Droste, Th. Jansen, and I. Wegener, A rigorous complexity analysis of the $(1+1)$ evolutionary algorithm for linear functions with Boolean inputs, In [41], pp. 499-504.

[35] S. Droste, Th. Jansen, and I. Wegener, On the optimization of unimodal functions with the $(1+1)$ evolutionary algorithm, In [42], pp. 47-56.

[36] Th. Jansen and I. Wegener, On the analysis of evolutionary algorithms – A proof that crossover really can help, In *European Symp. Algorithms*, Prague, July 1999, (in press).

[37] D. H. Wolpert and W. G. Macready, No free lunch theorem for optimization, *IEEE Trans. Evolutionary Computation*, Vol. 1, pp. 67-82, 1997.

[38] S. Droste, Th. Jansen, and I. Wegener, Perhaps not a free lunch but at least a free appetizer, In W. Banzhaf et al., Eds., *Proc. Genetic and Evolutionary Computation Conf.*, Morgan Kaufmann, San Francisco CA, 1999, (in press).

[39] G. Rudolph, On a multi-objective evolutionary algorithm and its convergence to the Pareto set, In [41], pp. 511-516.

[40] G. Winter, J. Périaux, M. Galán, and P. Cuesta, Eds., *Genetic Algorithms in Engineering and Computer Science*, Wiley, Chichester, 1995.

[41] D. B. Fogel, H.-P. Schwefel, Th. Bäck, and X. Yao, Eds., *Proc. Fifth IEEE Conf. Evolutionary Computation*, IEEE Press, Piscataway NJ, 1998.

[42] A. E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds., *Parallel Problem Solving from Nature – PPSN V*, Springer, Berlin, 1998, LNCS 1498.

[43] H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., *Parallel Problem Solving from Nature – PPSN IV*, Springer, Berlin, 1996, LNCS 1141.