communications in mathematical
match
and in computer chemistry

no. 38, October 1998

ISSN 0340-6253

MATCDY (38) 85-98 (1998)

Some Current Trends in Evolutionary Algorithm Research Exemplified by Applications in Computer-Aided Molecular Design

David E. Clark

Computer-Aided Drug Design, Rhone-Poulenc Rorer Ltd., Rainham Road South, Dagenham, Essex, RM10 7XS, United Kingdom (Email: david-e.clark@rp-rorer.co.uk)

Abstract

From its beginnings in the 1950s, the field of evolutionary algorithm (EA) research has grown dramatically, with much of the expansion occurring in the last decade. Many application areas have benefited from the ideas emerging from the evolutionary algorithm community, not the least of which is the field of computer-aided molecular design (CAMD). This short review presents some of the current areas of research in the field of EAs with examples applications drawn from CAMD. Among the topics covered are: the use of domain-specific knowledge to guide the choice of representation of population members and the implementation of evolutionary operators; the application of self-adaptive techniques; and the benefits of parallel and hybrid variations of evolutionary algorithms.

1 Introduction

The genesis of evolutionary computation can be traced back to the late 1950s, although it has only been in the last decade or so that the philosophy and methods it comprises have been widely disseminated and applied amongst the general scientific community [1]. At the time of writing, there can be few areas of scientific research involving problems of search, optimization or classification that remain untouched by the explosive growth of interest in what have become known as "evolutionary algorithms" (EAs).

A good example of this rapid surge of interest in EAs is to be found in the field of Computer-Aided Molecular Design (CAMD). A survey by the author of the number of publications in this field involving EAs over the past few years shows a remarkable increase (see Figure 1 - note that the 1997 figure is not for a complete year). These findings have been corroborated by Milne, who states in a recent article that, between 1989 and 1992, there were only 5 papers published with a chemical orientation that employed evolutionary algorithms; since 1993, however, that figure has mushroomed to 210 [2].

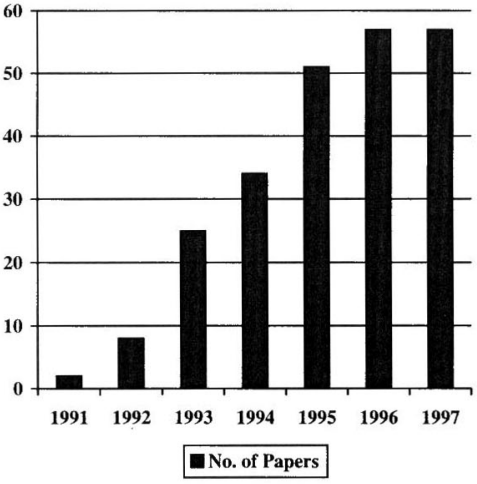

Figure 1: Graph of the number of publications in the last 7 years involving evolutionary algorithms in the field of Computer-Aided Molecular Design

To facilitate later discussions, a brief introduction to EAs will now be given. As the name suggests, evolutionary algorithms are inspired by the principles and processes observed in natural selection. As an illustration of this, an outline of a general evolutionary algorithm (following [1]) is shown in Figure 2:

```
t := 0;
initialize P(t);
evaluate P(t);
while not terminate do
        P′(t) := variation(P(t));
        evaluate(P′(t));
        P(t+1) := select(P′(t) ∪ Q);
        t := t+1;
enddo
```

Figure 2: An outline of a general evolutionary algorithm [1]

In this scheme, P(t) denotes a population of μ individuals at generation t. Each of the members of the population represents (or encodes) a trial solution to the problem in hand. The algorithm commences by initializing the population, usually with random values. Each of the population members is then assigned a fitness score using an objective function that assesses the "goodness" of the solution the individual represents. The population of offspring P′(t) containing λ members is then generated by the application of variation operators to the parent population P(t). Typically, the variation operators used are mutation (in which a random perturbation is applied to an individual) and/or crossover (in which two parent individuals combine to generate two offspring). The offspring population members are assessed using the fitness function and then a selection process takes place to determine the members of the next generation P(t+1). In Figure 2, Q denotes a special set of individuals that might be considered for selection. Possible values for Q could be Q=P(t) or Q=∅. The selection procedure is based upon the fitness of the individuals available for selection and is done in such a manner as to promote the chances of survival of fitter individuals. In this way, the algorithm proceeds towards better and better solutions until some termination criterion is attained. Varying the value of λ allows the formulation of steady-state (λ=1) or generational (λ=μ) evolutionary algorithm variants [1].

Historically, three classes of evolutionary algorithm have been distinguished based upon the manner in which individuals are represented, the type of variation operator(s) applied to the population and the way in which selection is accomplished. These three types are: genetic algorithms (GAs), of which genetic programming (GP) may be considered a subset, evolutionary programming (EP) and evolution strategies (ESs) [3]. However, it is generally recognised that the traditional barriers differentiating these classes are being rapidly broken down as researchers combine ideas from each to develop customized and improved algorithms. Today, many prefer to classify all evolution-inspired algorithms under the general heading of "evolutionary algorithms" or "evolutionary computation" [4].

It is not the purpose of this article to provide a detailed review of the applications of evolutionary algorithms in Computer-Aided Molecular Design - a number of such reviews exist for consultation by the interested reader [5-9]. Rather, a perspective is sought from the point of view of the EA community, examining some of the current trends and issues in EA research and illustrating them with applications from the field of CAMD.

2. Current Areas of Research in Evolutionary Algorithms

In what follows, a number of areas of research within the field of evolutionary algorithms will be highlighted. The list is not exhaustive but it does cover most of the interesting developments that are beginning to find their way into CAMD applications. A more thorough discussion of many of the current research areas within evolutionary computation can be found in an excellent and comprehensive publication that has appeared recently [10].

2.1 Basic Theory

One of the most pressing needs at the present time within evolutionary computation is for a more secure theoretical framework to guide users of current applications and upon which to base future research. As Baeck *et al.* remark: "We know that [evolutionary algorithms] work, but we do not know why" [1]. A major consequence of this lack of theory is that there are few rules of thumb to guide the design and parameterization of evolutionary algorithms. This point is expanded upon by Fogel who, while highlighting the limited usefulness of current theoretical analysis (such as the "schema theorem" [11]) for providing practical guidance in devising parameter settings or conditions for improving EA performance, appears somewhat gloomy about the prospects for progress in this area: "Although the hope of improving evolutionary algorithms in general function optimization on the basis of mathematical theory appears dim, such results would be most valuable" [12].

From a CAMD viewpoint, there is probably little that can be contributed directly to the solution of this dilemma, although the types of problem encountered routinely in molecular design and recognition studies can provide a challenging testbed for any new theoretical proposals from EA researchers. Close collaboration between the two communities of scientists should be mutually beneficial in this regard, a point that will be touched on again later.

2.2 Domain-specific Representation and Operators

For any evolutionary algorithm, the representation (or encoding) of the individuals comprising the evolving solution population and the set of operators applied to them to generate offspring are perhaps the two most important components of the system, in many cases being crucial in determining the success or failure of the algorithm. Clearly, there is a strong link between the representation chosen and the operators that can be employed (or *vice versa*) and so considerations about one will inevitably affect the choices made about the other [13]. Traditionally, evolutionary algorithms have represented the individuals comprising the evolving solution population by binary strings (GAs) or vectors of real values (EP, ESs) and the archetypal variational operators have been developed with these in mind. However, in more recent times, there has been much enthusiasm for the employment of more "natural", or problem-specific, representations (and thus, operators). In general, the guiding principle today seems to be "mould the algorithm to the problem, not the problem to the algorithm" [14,15].

There are at least two reasons for this. Firstly, as evolutionary algorithms have been applied to a wider range of problems, it has been increasingly found that many do not map well into a binary string- or vector-based representation. Thus, workers have been driven to experiment with more natural representations and operators and have often found success by doing so. Secondly, and more fundamentally, the recent "No Free Lunch" theorems of Wolpert and Macready [16] state broadly that, for any algorithm, any elevated performance on one class of problems is exactly countered by poorer performance on another class. Thus, an evolutionary algorithm tailored for a specific application by the incorporation of problem-specific knowledge in the encoding and operators is likely to outperform a canonical, "black-box" implementation.

There are a number of examples from the molecular design literature that illustrate the points made above. In developing a GA for protein folding simulations, Moult and Unger opted not to use any form of encoding of population members but rather to allow genetic operators to act directly upon the conformations of the protein model [17]. In the area of *de novo* molecular design, both Westhead *et al.* [18] and Glen and Payne [19] adopted a similar philosophy, having the evolving molecular structures as the population members and developing crossover and mutation operators to suit the problem representation. In the latter work, two different types of crossover and no fewer than twelve different mutation operators were employed for the manipulation and variation of molecular structures.

2.3 Self-Adaptation

One of the major difficulties in using an evolutionary algorithm is in deciding on the settings for the (many) adjustable parameters that are associated with this class of algorithm. Thus, another key area for evolutionary computation research identified in [1,12] is that of *self-adaptation*.

Most research on self-adaptation has emerged from the ES and EP communities and has thus been primarily concerned with the mutation operator, which is the main variational operator for these two classes of EA. The driving force for developing self-adaptive techniques was the realisation that, given a population of real-valued vectors, the optimization performance of the algorithm can be improved by applying perturbations of different magnitude to each of the variables in the vector, i.e., each dimension of the search. This is particularly so when the variables have different units of dimension, e.g. pressure and temperature [20]. To try to determine *a priori* optimal values for these mutational step sizes is virtually impossible, especially as the optimal values may alter during the course of a search as the algorithm traverses the fitness landscape. Self-adaptation seeks to counter this problem by allowing the mutational step sizes to adapt themselves as the evolutionary search progresses.

Two main methods for self-adaptation have been proposed: that due to Schwefel [21] and another independently developed by Fogel [22]. Of these, the former has become more widely accepted as it has demonstrated a generally superior optimization performance across a series of standard test functions [20,23]. The method of Schwefel is as follows: Each of the real-valued vectors of variables x comprising the population is given an accompanying vector of *strategy parameters*, σ, where $\sigma(\mathrm{i})$ denotes the standard deviation to use when applying a zero-mean Gaussian mutation to component $x(\mathrm{i})$ of the parent vector. Then:

$$\sigma'(\mathrm{i}) = \sigma(\mathrm{i}) \exp(\tau_0 \mathrm{N}(0,1) + \tau \mathrm{N}(\mathrm{i})(0,1))$$

and

$$x'(\mathrm{i}) = x(\mathrm{i}) + \mathrm{N}(0,\sigma'(\mathrm{i}))$$

where the constant $\tau = 1/[2(n^{1/2})]^{1/2}$, $\tau_0 = 1/(2n)^{1/2}$, N(0,1) is a standard Gaussian random variable sampled once for all n dimensions and N(i)(0,1) is a standard Gaussian random variable sampled anew for each of the n dimensions [20]. Thus, under the Schwefel scheme, at each generation, the strategy parameters for the individual are mutated and the new values are used to generate the offspring (the "sigma-first" method [24]).

Two groups of CAMD researchers have experimented with self-adaptive mutation in the context of using EP algorithms for the conformationally flexible docking of ligands to proteins. In the development of their EPDOCK program [24-26], Gehlhaar and co-workers have used both the sigma-first method outlined above and a variation they term "sigma-last" in which the parent σ values are first used to create offspring and then mutated. They found that the sigma-first method was superior probably because the offspring positions in the search space are determined by the offspring strategy parameters. This avoids the situation that can occur with the sigma-last method whereby offspring may be generated that have useful position vectors but poor strategy parameters [24]. Westhead *et al.* [27] also employed a sigma-first method but in their work found that better results were obtained using Cauchy, rather than Gaussian, mutation to perturb the x vectors - a tactic suggested by Yao and Liu [28].

According to Michalewicz [4], some attempts have been made to develop a self-adaptive form of the crossover operator, whereby the algorithm retains a record of the crossover points used and the fitness of the resulting individuals but this does not seem to have been widely adopted.

It is also worth briefly mentioning an alternative approach to the problem of setting parameter values - that of *meta-evolution*. In meta-evolution, one EA controls a population of other evolutionary algorithms each of which is initiated with different parameter settings. As this population of algorithms operates on the problem in hand, it becomes apparent over time which are performing well and thus, what are good sets of parameters. This type of method is being pioneered in programs such as DAGA-2 [29]. Such an approach also lends itself naturally to parallel architectures, of which more will be said in the next section.

2.4 Parallel Algorithms

In the execution of an evolutionary algorithm, it is invariably the fitness evaluation step that is the most computationally expensive. However, the fact that this step is decoupled from the rest of the algorithm makes it highly amenable to parallelization, creating a parallel evolutionary algorithm. The various kinds of parallel EA have been classified by Cantu-Paz [30]. For the purposes of this review, two types of parallel EA are of particular interest.

The first kind is what Cantu-Paz terms "coarse-grained" parallelism. In this formalism, the population is divided into a small number of subpopulations which are kept relatively isolated from one another, evolving on separate processors or machines. The introduction of a *migration* operator permits the exchange of individuals between subpopulations at specified intervals. At least two models of coarse-grained parallelism are possible: the first is termed the "island model" and the second is known as the "stepping stone" model. Both partition the population into subpopulations in the same way, the difference between them being that, while the island model allows migration between any two subpopulations, the stepping stone model allows migration only between neighbouring subpopulations [30]. When using coarse-grained parallelism, there is an additional benefit on top of the speedup gained from distributing the fitness calculations over a number of processors. The partitioning of the population into subpopulations may also help to maintain genetic diversity in the population as a whole. This can help search efficiency by ensuring good sampling of the search space and may also help prevent premature convergence to suboptimal solutions. This benefit of coarse-grained parallelism can be taken advantage of even on serial processors, as will be demonstrated below. A thorough discussion of island models can be found in [31].

The second type of interest for the purposes of this review is "fine-grained" parallelism. In this kind of parallel EA, the population is subdivided into a large number of very small subpopulations. In the limiting (and ideal) case, each individual is assigned to its own processing element in a massively parallel computer [30].

Both types of parallelism have been experimented with in CAMD applications. The island model has been successfully employed by Jones and co-workers in developing genetic algorithms for protein-ligand docking [32] and molecular superposition and pharmacophore identification [33]. These algorithms were run on a serial machine with the subpopulations residing on the same processor. Nonetheless, in experiments comparing the use of a single population of 500 individuals to the use of 5 subpopulations of 100 individuals, it was found that the island model gave equivalent results in slightly shorter run times [32]. This would seem to indicate that the maintenance of genetic diversity through distributed populations can aid search efficiency as well as the quality of the final solution. Beckers *et al.* obtained a similar result using a stepping stone model in their GA for structure determination from NMR spectra [34]. Their parallel GA was run over a local area network of workstations and significant speedups compared to a sequential implementation were observed arising from both the parallelization of the fitness evaluation and the fact that the parallel runs required fewer fitness evaluations to reach convergence (indicating a more efficient search). A stepping stone model was also used by Del Carpio in work on protein folding in which a network of five transputers was employed [35].

There have been fewer applications of massively parallel evolutionary algorithms in CAMD, presumably because of a lack of suitable machines. However, Shapiro and Wu have developed a GA for RNA folding predictions that runs on a MasPar MP-2 16384-processor machine [36]. This GA begins by initialising a population of simple RNA structures, one for each processor. At each generation, for each processor, the GA selects two parents from the structures stored on the processor in question and its eight neighbours. This step takes advantage of the eight-way interconnected mesh structure of the MasPar machine. Mutation and crossover operations are then performed to generate two child structures and the best of these replaces the current structure for the particular processor. All these operations take place in parallel, generating 16384 new structures at each generation [36].

Finally, Wild and Willett discuss a number of different models of parallelism in the context of a GA for similarity searching in databases of 3-D chemical structures [37]. In their work, a Kendall Square Research KSR-1 machine with 64 processors was used and a simple strategy of assigning single molecules to single processors was found to be the most successful.

2.5 Hybrid Algorithms

While evolutionary algorithms can often yield excellent results when applied alone to a problem, additional benefits can often be derived from their combination with other computational methods. There is often considerable synergy to be exploited between evolutionary algorithms and optimization methods from mathematical programming, greedy or local search algorithms or other heuristic search algorithms such as simulated annealing (SA) and tabu search (TS) [38]. Evolutionary algorithms have also been successfully hybridized with neural and fuzzy computing methods [39,40].

Given that evolutionary algorithms are often viewed as global optimization methods, one of the most common hybridizations is with a local optimization method. This has been frequently employed in CAMD applications. Gehlhaar *et al.* [25] and Westhead *et al.* [27] employed the Powell minimization algorithm [41] to refine the final solutions generated by evolutionary docking algorithms. In studies of the conformational search of large molecules, McGarrah and Judson found that frequent gradient optimization of the conformational energy during the course of a GA search gave a marked advantage over methods in which the energy was optimized by the GA alone [42]. However, in studies on molecules having smaller conformational search spaces, the same workers indicate that the GA was capable of locating good solutions without the computationally expensive optimization step [43].

Neural networks have been usefully combined with evolutionary algorithms for the generation of Quantitative Structure-Activity Relationships (QSARs). The most advanced example of this is the Genetic Neural Networks (GNN) method of So and Karplus [44,45]. GNN employs an EP for feature selection and a neural network for the generation of coefficients for the equation relating the feature values of the molecules under study to their observed biological activities. Other work in this area is reported in [46].

More recently, a K-nearest-neighbours (KNN) classification algorithm has been hybridized with a GA to form the CONSOLV program developed by Raymer *et al.* [47]. This program derives and applies rules to decide whether a given water molecule in a protein active site is likely to be involved in or displaced by ligand binding. The same group of workers have also experimented with a KNN-genetic programming hybrid and found it to give superior results to the GA variant [48].

Finally, while there have been no fully published accounts in CAMD of the hybridization of an evolutionary algorithm with another heuristic search algorithm, two groups of workers have hinted that, in the context of ligand-protein docking, such hybrid algorithms (GA/TS [27], EP/SA [49]) can yield superior performance to either of the algorithms used in isolation.

2.6 Collaborations and Commercial Applications

Two final trends that can be identified in relation to evolutionary algorithms and CAMD (and which may be true for other application domains as well) are firstly, the increase in collaborative research between EA experts and CAMD scientists and secondly, the rise in the number of commercial software products incorporating EA methods.

In terms of research collaborations, two fruitful examples are the pairing of scientists from Natural Selection Inc. and Agouron Pharmaceuticals Inc. in the development of the EPDOCK program [24-26], and the combined efforts of researchers from the Computer Science and Biochemistry Departments of Michigan State University in the creation of CONSOLV [47,48]. In both cases, the former groups supply EA expertise while the latter bring in-depth knowledge of the problem in hand. As evolutionary algorithm methodologies continue to develop and as CAMD researchers seek to gain maximum value from their EA applications, such collaborations are likely to become more common bringing benefit to both communities.

It is no great surprise that EA applications have not taken long to appear in commercial CAMD software - one of the great strengths of evolutionary algorithms is that they represent a methodological framework that is easy to understand and work with. Commercial programs using evolutionary algorithms or principles are now available for *de novo* molecular design, ligand-protein docking, molecular superposition and pharmacophore identification [50] and QSAR [51]. Given the large rise in CAMD applications of evolutionary algorithms (see Figure 1), it is likely that the number of commercial applications will continue to increase.

3 Future Directions

The sections above have attempted to delineate some of the current developments in EA research and to show how they have often been very rapidly taken up by applications in CAMD. The field of evolutionary computation is still a young one and, as Baeck *et al.* note: "...we are convinced that we are just beginning to understand and to exploit the full potential of evolutionary computation" [1]. In looking to the future, there are a number of promising research directions that have been identified by EA practitioners as being likely to receive much attention. One of these is the further development of the most recent class of EA, genetic programming, which is concerned with the evolution of computer programs for the (approximate) solution of problems [52]. This field has grown rapidly since its inception in the early 1990s and some applications of relevance to CAMD are beginning to emerge [48,53,54]. Directions for future work in GP research are presented by Koza [55]. More generally, workers are keen to try to incorporate into evolutionary computation more of the as yet untapped phenomena and mechanisms observed in organic evolution, in the hope that this will lead to more useful and robust algorithms [1,12,56]. Ideas such as including individuals of different gender in the evolving population and applying operators and selection rules differently to different sexes, or taking account of co-evolutionary phenomena are being considered with the aim of making evolutionary computation a more faithful mimic of its organic analogue.

A quote from [56] provides an excellent close to this section: "Many more ideas for improving EAs may come forth when biologists and computer scientists or engineers/managers using EAs in design, control, and management or other decision-making processes, sit together without prejudice concerning the scholarliness of their points of view. Obstacles to further success do not lie in the real world; they lie only in the heads of those who try to stay self-contained within their narrow single disciplines. The field of 'directions for future research', in principle, is wide open and - certainly - full of chance and surprises".

4 Conclusions

The field of evolutionary computation has already had a significant impact upon computer-assisted molecular design. (An extensive bibliography of evolutionary algorithms applied to CAMD can be obtained from the author in either Microsoft Word or HTML format). In this article, some of the current trends discernible in EA research have been reviewed and illustrated by applications from the realm of CAMD. There appears to be great potential for developing improved evolutionary algorithms through interdisciplinary collaboration and research; this promises even more powerful and robust applications for CAMD scientists in the future.

Acknowledgements

I should like to thank the editors for inviting me to contribute this article and also my colleagues in the Computer-Aided Drug Design group at Rhone-Poulenc Rorer, Drs. Richard Lewis, Stephen Pickett and Paul Bamborough, for reading and commenting upon the manuscript while it was in preparation.

References

[1] BAECK, T., HAMMEL, U., AND SCHWEFEL, H.-P. Evolutionary Computation: Comments on the History and Current State. *IEEE Trans. Evol. Comput.*, **1997**, 1, 3-17.

[2] MILNE, G.W.A. Mathematics as a Basis for Chemistry. *J. Chem. Inf. Comput. Sci.*, **1997**, 37, 639-644.

[3] BAECK, T. AND SCHWEFEL, H.-P. An Overview of Evolutionary Algorithms for Parameter Optimisation. *Evol. Comput.*, **1993**, 1, 1-23.

[4] MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, 1996.

[5] JUDSON, R. Genetic Algorithms and Their Use in Chemistry. *Rev. Comput. Chem.*, **1997**, 10, 1-73.

[6] MADDELENA, D.J. AND SNOWDON, G.M. Applications of Genetic Algorithms to Drug Design. *Exp. Opin. Ther. Patents*, **1997**, 7, 247-254.

[7] PARRILL, A. Evolutionary and Genetic Methods in Drug Design. *Drug Discovery Today* **1996**, 1, 514-521.

[8] DEVILLERS, J., editor, *Genetic Algorithms in Molecular Modelling*, Academic Press, New York, 1996.

[9] CLARK, D.E. AND WESTHEAD, D.R. A Review of Evolutionary Algorithms in Computer-Aided Molecular Design. *J. Comput.-Aided Mol. Des.*, **1996**, 10, 337-358.

[10] BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[11] GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*, Chapter 2, Addison-Wesley, Reading (MA), 1989.

[12] FOGEL, L.J. Future Research in Evolutionary Computation. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section H1.2, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[13] MICHALEWICZ, Z. Introduction to Search Operators. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section C3.1, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[14] LUKE, B.T. An Overview of Genetic Methods. In DEVILLERS, J., editor, *Genetic Algorithms in Molecular Modelling*, 35-66, Academic Press, 1996.

[15] FOGEL, D.B. AND ANGELINE, P.J. Guidelines for a Suitable Encoding. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section C1.7, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[16] WOLPERT, E.D.H. AND MACREADY, W.G. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.*, **1997**, 1, 67-82.

[17] UNGER, R. AND MOULT, J. Genetic Algorithms for Protein Folding Simulations. *J. Mol. Biol.*, **1993**, 231, 75-81.

[18] WESTHEAD, D.R., CLARK, D.E., FRENKEL, D., LI, J., MURRAY, C.W., ROBSON, B. AND WASZKOWYCZ, B. PRO_LIGAND: An Approach to De Novo Molecular Design. 3. A Genetic Algorithm for Structure Refinement. *J. Comput.-Aided Mol. Des.*, **1995**, 9, 139-148.

[19] GLEN, R.C. AND PAYNE, A.W.R. A Genetic Algorithm for the Automated Generation of Molecules within Constraints. *J. Comput.-Aided Mol. Des.*, **1995**, 9, 181-202.

[20] FOGEL, D.B. Mutation: Real-valued Vectors. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section C3.2.2, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[21] SCHWEFEL, H.-P. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[22] FOGEL, D.B., FOGEL, L.J. AND ATMAR, J.W. Meta-evolutionary Programming. In CHEN, R.R., editor, *Proceedings of the 25th Asilomar Conference on Signals, Systems and Computers*, 540-545, Maple Press, San Jose (CA), 1991.

[23] SARAVANAN, N., FOGEL, D.B. AND NELSON, K.M. A Comparison of Methods for Self-Adaptation in Evolutionary Algorithms. *BioSystems*, **1995**, 36, 157-166.

[24] GEHLHAAR, D.K. AND FOGEL, D.B. Tuning Evolutionary Programming for Conformationally Flexible Molecular Docking. In FOGEL, L.J., ANGELINE, P.J. AND BAECK, T., editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, 419-429, MIT Press, Cambridge (MA), 1996.

[25] GEHLHAAR, D.K., VERKHIVKER, G.M., REJTO, P.A., SHERMAN, C.J. FOGEL, D.B., FOGEL, L.J. AND FREER, S.T. Molecular Recognition of the Inhibitor AG-1343 by HIV-1 Protease: Conformationally Flexible Docking by Evolutionary Programming. *Chem. Biol.*, **1995**, 2, 317-324.

[26] GEHLHAAR, D.K., VERKHIVKER, G.M., REJTO, P.A., FOGEL, D.B., FOGEL, L.J. AND FREER, S.T. Docking Conformationally Flexible Small Molecules into a Protein Binding Site Through Evolutionary Programming. In McDONNELL, J.R., REYNOLDS, R.G. AND FOGEL, D.B., editors, *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, 615-627, MIT Press, Cambridge (MA), 1995.

[27] WESTHEAD, D.R., CLARK, D.E. AND MURRAY, C.W. A Comparison of Heuristic Search Algorithms for Molecular Docking. *J. Comput.-Aided Mol. Des.*, **1997**, 11, 209-228.

[28] YAO, X. AND LIU, Y. Fast Evolutionary Programming. In FOGEL, L.J., ANGELINE, P.J. AND BAECK, T., editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, 257-266, MIT Press, Cambridge (MA), 1996.

[29] WANG, G., DEXTER, T.W. AND PUNCH, W.F. Optimization of a GA and Within a GA for a 2-Dimensional Layout Problem. In GOODMAN, E., PUNCH, W.F., USKOV, V., editors, *Proceedings of the First International Conference on Evolutionary Computation and Its Applications*, 18-29, Russian Academy of Sciences, 1996.

[30] CANTU-PAZ, E. A Summary of Research on Parallel Genetic Algorithms. *IlliGAL Report No. 95007*, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana (IL), 1997.

[31] MARTIN, W.N., LIENIG, J. AND COHOON, J.P. Island (Migration) Models: Evolutionary Algorithms Based on Punctutated Equilibria. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section C6.3, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[32] JONES, G., WILLETT, P., GLEN, R.C., LEACH, A.R. AND TAYLOR, R. Development and Validation of a Genetic Algorithm for Flexible Docking. *J. Mol. Biol.*, **1997**, 267, 727-748.

[33] JONES, G., WILLETT, P. AND GLEN, R.C. A Genetic Algorithm for Flexible Molecular Overlay and Pharmacophore Elucidation. *J. Comput.-Aided Mol. Des.*, **1995**, 9, 532-549.

[34] BECKERS, M.L.M., DERKS, E.P.P.A., MELSSEN, W.J. AND BUYDENS, L.M.C. Parallel Processing of Chemical Information in a Local Area Network. III. Using Genetic Algorithms for Conformational Analysis of Biomacromolecules. *Comput. Chem.*, **1996**, 20, 449-457.

[35] DEL CARPIO, C.A. A Parallel Genetic Algorithm for Polypeptide Three-Dimensional Structure Prediction: A Transputer Implementation. *J. Chem. Inf. Comput. Sci.*, **1996**, 36, 258-269.

[36] SHAPIRO, B.A. AND WU, J.C. Predicting RNA H-Type Pseudoknots with the Massively Parallel Genetic Algorithm. *CABIOS*, **1997**, 13, 459-471.

[37] WILD, D.J. AND WILLETT, P. Similarity Searching in Files of Three-Dimensional Chemical Structures: Alignment of Molecular Electrostatic Potential Fields with a Genetic Algorithm. *J. Chem. Inf. Comput. Sci.*, **1996**, 36, 159-167.

[38] IBARAKI, T. Combinations with Other Optimization Methods. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section D.3, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[39] PORTO, W.V. Neural-Evolutionary Systems. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section D.1, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[40] KARR, C.L. Fuzzy-Evolutionary Systems. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section D.2, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[41] POWELL. M.J.D. A Restart Procedure for the Conjugate Gradient Method. *Mathematical Programming*, **1977**, 12, 241-254.

[42] McGARRAH, D.B. AND JUDSON, R.S. Analysis of the Genetic Algorithm Method of Molecular Conformation Determination. *J. Comput. Chem.*, **1993**, 14, 1385-1395.

[43] JUDSON, R.S., JAEGER, E.P., TREASURYWALA, A.M. AND PETERSON, M.L. Conformational Searching Methods for Small Molecules. II. Genetic Algorithm Approach. *J. Comput. Chem.*, **1993**, 14, 1407-1414.

[44] SO, S.S. AND KARPLUS, M. Evolutionary Optimization in Quantitative Structure-Activity Relationship: An Application of Genetic Neural Networks. *J. Med. Chem.*, **1996**, 39, 1521-1530.

[45] SO, S.S. AND KARPLUS, M. Genetic Neural Networks for Quantitative Structure-Activity Relationships: Improvements and Application of Benzodiazepine Affinity for Benzodiazepine/GABA(A) Receptors. *J. Med. Chem.*, **1996**, 39, 5246-5256.

[46] KYNGAS, J. AND VALJAKKA, J. Evolutionary Neural Networks in Quantitative Structure-Activity Relationships of Dihydrofolate Reductase Inhibitors. *Quant. Struct.-Act. Relat.*, **1996**, 15, 296-301.

[47] RAYMER, M.L., SANSCHAGRIN, P.C, PUNCH, W.F., VENKATARAMAN, S., GOODMAN, E.D. AND KUHN, L.A. Predicting Conserved Water-Mediated and Polar Ligand Interactions in Proteins Using a K-nearest-neighbors Genetic Algorithm. *J. Mol. Biol.*, **1997**, 265, 445-464.

[48] RAYMER, M.L., PUNCH, W.F., GOODMAN, E.D. AND KUHN, L.A. Genetic Programming for Improved Data Mining: Application to the Biochemistry of Protein Interactions. In KOZA, J.R., GOLDBERG, D.E., FOGEL, D.B. and RIOLO, R.L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 375-380, MIT Press, Cambridge (MA), 1996.

[49] BOUZIDA, D., GEHLHAAR, D.K., REJTO, P.A., VERKHIVKER, G.M. AND FREER, S.T. Efficient Configurational Search Methods for Flexible Ligand Docking. *Abstracts of the 11th European Symposium on Quantitative Structure-Activity Relationships: Computer-Assisted Lead Finding and Optimization*, P-37.D, Lausanne, 1996.

[50] LeapFrog, FlexiDock, GASP. Tripos Inc., 1699 S. Hanley Road, St. Louis (MO), USA.

[51] C^2.GFA. Molecular Simulations Inc., 9685 Scranton Road, San Diego (CA), USA.

[52] BANZHAF, W., NORDIN, P., KELLER, R.E. AND FRANCONE, F.D. *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Applications*, Morgan Kaufmann, San Francisco, 1997.

[53] HANDLEY, S. Automated Learning of a Detector for Alpha-Helices in Proteins via Genetic Programming. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, 271-278, Morgan Kaufmann, San Mateo (CA), 1993.

[54] KOZA, J.R. Classifying Protein Segments as Transmembrane Domains Using Genetic Programming and Architecture-Altering Operations. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section G6.1, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[55] KOZA, J.R. Future Work and Practical Applications of Genetic Programming. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section H1.1, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

[56] SCHWEFEL, H.-P. Challenges to and Future Developments of Evolutionary Algorithms. In BAECK, T., FOGEL, D.B. AND MICHALEWICZ, Z., editors, *Handbook of Evolutionary Computation*, Section H1.3, IOP Publishing and Oxford University Press, Bristol/New York, 1997.