Genetic and Evolutionary Algorithms

Gareth Jones
University of Sheffield, UK

Abbreviations

EA = evolutionary algorithm; EP = evolutionary programming; ES = evolution strategies; GA = genetic algorithm; GFA = genetic function approximation.

1 INTRODUCTION

Many applications in computational chemistry have a search space that is exponentially proportional to the problem dimensions and but for the simplest of cases these problems cannot be solved using exhaustive search methods. Consequently, there is considerable interest in heuristic techniques that attempt to discover near-optimal solutions within an acceptable time. Genetic algorithms (GAs) and other related *evolutionary algorithms* (EAs) provide a framework for effectively sampling large search spaces, and the basic technique is both broadly applicable and easily tailored to specific problems (see ***Genetic Algorithms: Introduction and Applications***). All that is required to apply an EA to any particular problem is an appropriate encoding scheme and a target function. Since 1992 we have seen an explosion in the number of seemingly intractable problems to which EAs have been successfully applied. This article reviews the application of EAs within the field of computational chemistry.

2 EVOLUTIONARY ALGORITHMS

EAs are computer programs that attempt to solve complex problems by mimicking the processes of Darwinian evolution.[1] In an EA a number of artificial creatures search over the space of the problem. They compete continually with each other to discover optimal areas of the search space. It is hoped that over time the most successful of these creatures will evolve to discover the optimal solution.

The artificial creatures in EAs, known as individuals, are typically represented by fixed length strings or vectors. Each individual encodes a single possible solution to the problem under consideration. For example, in order to construct an EA to search the conformation space of a molecule, each angle of rotation around a flexible bond could be encoded as a real number. Concatenating these numbers gives a string which can be used within an EA. Thus, each individual would encode a specific set of torsion angles. EAs manipulate pools or populations of individuals. The EA is started with an initial population of size μ comprising random individuals (that is, each value in every string is set using a random number generator). Every individual is then assigned a fitness value. To generate a fitness score the individual is decoded to produce a possible solution to the problem. The value of this solution is then calculated using the fitness function. Population members with high fitness scores therefore represent better solutions to the problem than individuals with lower fitness scores. Following this initial phase the main iterative cycle of the algorithm begins. Using mutation (perturbation) and recombination operators, the μ individuals in the current population produce λ children. The λ children are assigned fitness scores. A new population of μ individuals is then formed from the μ individuals in the current population and the λ children. This new population becomes the current population and the iterative cycle is repeated. At some point in the cycle evolutionary pressure is applied. That is, the Darwinian strategy of the survival of the fittest is employed and individuals compete against each other. This is achieved by selection based on fitness scores, with fitter individuals more likely to be selected. The selection is applied either when choosing individuals to parent children or when choosing individuals to form a new population.

There have been three main independent implementation instances of EAs:[2–4] GAs, developed by Holland[5] and thoroughly reviewed by Goldberg;[6] evolution strategies (ESs), developed in Germany by Rechenberg[7] and Schwefel;[8] and evolutionary programming (EP), originally developed by L. J. Fogel et al.[9] and subsequently refined by D. B. Fogel.[4] Each of these three algorithms has been proved capable of yielding approximately optimal solutions given complex, multimodal, non-differential, and discontinuous search spaces. Success has also been achieved for noisy and time-dependent landscapes. A simple description of each technique is given here and more formal descriptions are given by Bäck and Schwefel,[2] Bäck,[3] and Fogel.[4]

2.1 Genetic Algorithms

Figure 1 shows the canonical GA as developed by Holland.[5] The canonical GA encodes the problem within binary string individuals. Evolutionary pressure is applied in step 3, where the stochastic technique of roulette wheel parent selection is used to pick parents for the new population. The concept is

1. A population of μ random individuals is initialized.
2. Fitness scores are assigned to each individual.
3. Using roulette wheel parent selection $\mu/2$ pairs of parents are chosen from the current population to form a new population.
4. With probability P_c, children are formed by performing crossover on the $\mu/2$ pairs of parents. The children replace the parents in the new population.
5. With probability P_m, mutation is performed on the new population.
6. The new population becomes the current population.
7. If the termination conditions are satisfied exit, otherwise go to step 3.

Figure 1 A canonical GA

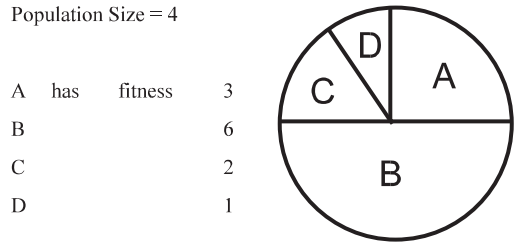

Figure 2 Roulette wheel parent selection

illustrated in Figure 2, using a trivial example with a population of four individuals. Each individual is assigned a sector of a roulette wheel that is proportional to its fitness and the wheel is spun to select a parent. While selection is random and any individual has the capacity to become a parent, selection is clearly biassed towards fitter individuals. Parents are not required to be unique and, in each iteration, fit individuals may produce many offspring.

From a population of size μ, $\mu/2$ pairs of parents are chosen. These parents form a new population. With probability P_c each pair is recombined using the crossover operator to produce a pair of children. This cut and splice operator is illustrated in Figure 3. A cross point is selected at random. Each child is identical to one parent before the cross point and identical to the other after the cross point. The child individuals then replace their parents in the new population. Following crossover, mutation is applied to all individuals in the new population. With probability P_m, each bit on every string is inverted. The new population then becomes the current population and the cycle is repeated until some termination criteria are satisfied. The algorithm typically runs for some fixed number of iterations, or until convergence is detected within the population. The probabilities of mutation and crossover, P_m and P_c are parameters of the algorithm and must be set by the user. Typical values are 0.8 for P_c and 0.01 for P_m, indicating that the algorithm is driven by recombination, and mutation is largely a background operator. Holland's schema theory provides a framework to explain how parts of the binary string encoding for high fitness multiply through iterations of the algorithm.[5]

Many GAs applied to real world problems bear only a passing resemblance to the canonical GA, and GAs are best viewed as a paradigm for evolutionary search, rather than a specific algorithm. The binary encoding is often inappropriate for many problems and may be extended to nonbinary representations.[10] Successful GAs have used integer string individuals[6,11,12] or even more general representations such as tree and matrix structures.[13,14] Specialized crossover operators have been devised to handle unusual encodings: the Partially-Matched crossover operator recombines integer strings that are constrained not to contain any duplicates.[12] In order to increase program effectiveness hybrid GAs mix problem specific operations with crossover and mutation.[11]

Selection pressure is defined as the relative probability that the fittest individual in the population will be chosen as a parent relative to an individual of average fitness. Too high a selection pressure and a GA will rapidly converge to a suboptimal solution. While encouraging search, a low selection pressure can result in a GA taking an inordinate time to converge. In order to control selection pressure within a GA fitness values are often rescaled when applying roulette wheel parent selection.[6,11]

One problem with a canonical GA is that there is no guarantee that good individuals will survive from one iteration to the next. Not all algorithms produce an entirely new population each iteration. An elitist strategy involves copying the best individuals unchanged from the current population to the new population.[11] In a steady-state GA, each iteration involves the application of one crossover or mutation operator and only one or two new individuals are added to the population, usually replacing the worst individuals.[11,15] Davis has compiled a number of examples of the application of GAs to real world problems and the modifications to the canonical GA that were required.[11]

GAs have proved to be the most popular of the three EAs. They provide a simple framework for attempting to solve complex search problems and have been widely applied in computational chemistry.

2.2 Evolution Strategies

Figure 4 shows the ES as developed by Rechenberg[7] and Schwefel.[8] Historically ESs were designed for parameter optimization problems. The encoding used in an individual is therefore a list of real numbers: these are called the object variables of the problem. Additionally, each individual contains a number of strategy parameters, these being the variances and covariances of the object variables (the covariances are optional, but when used are normally defined using the rotation angles of the covariance matrix). The strategy parameters are used to control the behavior of the mutation operator and are not required when decoding an individual.

In each iteration λ offspring are generated from a population of size μ, where λ is typically 7μ. The recombination operator produces one child and requires two parents for each object variable and strategy parameter in the child. Historically, the same parents are used to generate all object variables in the child, then the parents are re-selected for each strategy parameter. The parents are selected randomly from the current population (i.e., there is no selection pressure at this point). A

Figure 3 The GA crossover operator

1. A current population of μ individuals is randomly initialized.
2. Fitness scores are assigned to each of the μ individuals.
3. λ new offspring are generated by recombination from the current population.
4. The λ new offspring are mutated.
5. Fitness scores are assigned to the λ new offspring.
6. A new population of μ individuals is selected, using either (μ, λ) or $(\mu + \lambda)$ selection.
7. The new population becomes the current population.
8. If the termination conditions are satisfied exit, otherwise go to step 3.

Figure 4 A simple ES

number of alternative recombination techniques are available, but the best results have been observed by setting each object variable in the child to be the same as the object variable in one of the parents and setting each strategy parameter in the child to be the mean of the parameter's values in the parents. Mutation is the main operator in the ES and acts upon strategy parameters as well as object variables. The mutation operator first perturbs the strategy parameters. The object variables are then mutated using the resulting probability distribution defined by the modified strategy parameters. This special mutation operator allows the ES to evolve good strategy parameters for the problem and has been termed self-adaptation. Selection in EAs is deterministic: the best μ individuals are taken from the λ new offspring ((μ, λ) selection) or from the union of μ parents and λ offspring ($(\mu + \lambda)$ selection). The preferred method is (μ, λ) selection, since $(\mu + \lambda)$ selection can disrupt the self-adaptation mechanism. Like the GA, EAs run until some termination criteria are satisfied. There exists a limited amount of convergence theory for ESs.[2]

As yet there is only one published application of ESs in the field of computational chemistry, that of Schneider et al.[16] However, in ESs remains an attractive alternative to GAs, especially in the field of parameter optimization, where in model systems they appear to outperform GAs.[2]

2.3 Evolutionary Programming

Figure 5 illustrates the form of an EP scheme. EP was originally developed by L. J. Fogel et al.[9] for the evolution of finite state machines using a limited symbolic alphabet encoding. Subsequently D. B. Fogel extended the EP to encode real numbers, thus providing a tool for variable optimization.[4] Individuals in the EP comprise a string of real numbers, as in ESs. EP differs from GAs and ESs in that there is no recombination operator. Evolution is wholly dependent on the mutation operator, which uses a Gaussian probability distribution to perturb each variable. The standard deviations correspond to the square root of a linear transform of the parents' fitness score (the user is required to parametrize this transform). To overcome parametrization problems associated with the linear transform Fogel developed meta-evolutionary programming (meta-EP).[17] In meta-EP individuals encode both object variables and variances (one variance for each object variable). As in ESs the variances are self-adapted and used to control the Gaussian mutation operator.

Selection pressure is applied in the EP when forming a new population from parents and offspring of the mutation operator, using a mechanism called tournament selection. Stochastic q-tournament selection is employed, where q is a parameter of the algorithm. Let U be the union of all parents and offspring. For each member m of U, q opponents are selected from U at random. A count is then made of the number of opponents that have worse fitness scores than m. The μ individuals with the highest tournament counts go on to form the new population. Note that as q increases the selection pressure in the algorithm increases and the selection process becomes increasingly deterministic. One side-effect of this selection process is that the best individual is always present in the new population.

1. A current population of μ individuals is randomly initialized.
2. Fitness scores are assigned to each of the μ individuals.
3. The mutation operator is applied to each of the μ individuals in the current population to produce μ offspring.
4. Fitness scores are assigned to the μ offspring.
5. A new population of size μ is created from the μ parents and the μ offspring using tournament selection.
6. If the termination conditions are satisfied exit, otherwise go to step 3.

Figure 5 A simple EP scheme

Like ESs, applications using EP are rare in computational chemistry, though some successes have been achieved by Luke[18] and Gehlhaar et al.[19] Like ESs, EP is a technique best suited to parameter optimization.

2.4 Summary

Three main types of EA have been developed: GAs, ESs, and EP. These algorithms are broadly similar, yet there are significant differences. All operate on fixed length strings, which contain real values in ESs and EP and binary numbers in the canonical GA. All algorithms incorporate a mutation operator: for ESs and EP mutation is the driving force. GAs and ESs also use a recombination operator, which is the primary operator for the GA. All three use a selection operator which applies evolutionary pressure, either extinctive (in ESs and EP, the operator determines which individuals will be excluded from the new population) or preservative (in the GA the operator selects individuals for breeding). In GAs and EP selection is probabilistic, while ESs use a deterministic selection. ESs and meta-EP allow self-adaptation, where parameters controlling mutation are allowed to evolve along with object variables. Finally, it is worth noting that the implementer is free to modify these algorithms. For example, the GA can be run using an integer alphabet.

3 APPLICATIONS OF EVOLUTIONARY ALGORITHMS IN COMPUTATIONAL CHEMISTRY

Over recent years the application of EAs to complex problems in computational chemistry has become commonplace and there is now a growing collection of published applications. Recent literature reviews include Clark and Westhead[20] on applications in computer-aided molecular design, Willett on molecular recognition,[21] Parrill on drug design,[22] Pedersen and Moult on protein structure prediction,[23] and Judson in chemistry.[24] Additionally, Devillers has edited a volume of applications in molecular modeling.[25]

3.1 Macromolecular Structure Prediction

The prediction of macromolecular structure and, particularly, protein tertiary structure (the 'folding' problem; see ***Protein Folding and Optimization Algorithms***) remains one of the most challenging problems in computational chemistry. Not only is the conformational space enormous, but the mechanisms of folding remain poorly understood. EAs have proved a popular approach to tackling the search space problem. As in all folding algorithms, considerable difficulty has been encountered in designing suitable fitness functions. Given that the success of any technique depends on the objective function it is thus difficult to evaluate and compare search strategies.

3.1.1 Protein Structure Prediction

(See ***Protein Structure Prediction in 1D, 2D, and 3D***.)

Numerous practitioners have used EAs to attempt to solve the folding problem. An early study suggested the superiority of GAs over simulated annealing by performing an idealized study on a 2D grid.[26] Bowie and Eisenberg also demonstrated the effectiveness of GAs by using an empirical guiding function.[27] A library of peptide fragment conformations was used to build the starting population and mutation and crossover were more likely to occur at fragment junctions.

Using a tetrahedral 3D lattice, Dandekar and Argos illustrated how GAs could fold a four β-strand bundle.[28] A binary encoding was used, where two bits encoded the direction of the main chain through the lattice from each Cα atom. A simple fitness function, which minimized clashes and scored for β-sheet formation, was employed. Later work has concentrated on grid-free experiments. Using a different binary encoding, where each residue is assigned a conformation from a small library (typically, the residue can be in one of seven conformations), Dandekar and Argos successfully folded small helical proteins.[29] The fitness function was more complex, being a linear sum of terms which scored for bad clashes, secondary structure formation, tertiary structure formation, hydrophobic burial, and hydrogen bonding. The potential function has been further refined for nonhelical[30] proteins and recent work shows good results on 19 test systems of mixed secondary structure type.[31] Sun and co-workers have also achieved considerable success in protein folding using a GA. Using a statistical potential in the fitness function, and a binary string representation, melittin and apamin were successfully folded.[32] Binary numbers were used to encode a state on the Ramachandran map for each residue. The formation of the initial population and the mutation operator was assisted by a dictionary of peptide segment conformations. More recently the algorithm has performed well on two test sets of ten small proteins.[33,34] In another GA study, Le Grand and Merz encoded protein torsional angles as binary numbers in each individual.[35] Unfortunately problems were encountered with the knowledge-based potential function, though the GA proved effective at searching conformational space.

The techniques described above usually require knowledge of protein secondary structure. However, Pedersen and Moult[36] have used a GA to successfully perform *ab initio* folding of a 22-residue peptide. The fitness function was parametrized by a potential of mean force analysis of experimental structures.

In order to guide GAs to solve this seemingly intractable problem some groups have devised specialist operators to be used with crossover and mutation within the GA. For example, Rabow and Scheraga use a Cartesian space operator for recombination,[37] while Elofsson et al. have devised a local operator to perturb small sections of the backbone.[38] In attempts to improve search methods other techniques have been combined with GAs: Gunn et al. have folded myoglobin using parallel simulated annealing with a mixing of states achieved by a GA;[39] Del Carpio performed local optimization of offspring following crossover and mutation.[40] Both of these implementations utilized parallel hardware.

Predicting the tertiary structure of part of the protein is in itself a demanding problem. Ring and Cohen have used a GA to sample the conformational space of loop regions.[41] An alphabet of tetrapeptide conformations was defined and each GA individual encoded conformations of partially overlapping tetrapeptides. Tufféry et al. have used a GA to predict side-chain packing.[42] A rotamer library was used to describe side-chain conformations and each GA individual encoded one rotamer for every side-chain. Molecular mechanics energy terms were used in the fitness function. Tufféry et al. have also compared the effectiveness of GAs and other techniques in side-chain packing.[43] While the GA was as effective as the other algorithms it was not the most efficient.

GAs have also been applied to the simpler problem of secondary structure prediction. Acceptable results have been obtained by Vivarelli et al. using a neural network supervised by a GA.[44]

3.1.2 DNA Structure Prediction

Lucasius et al. have developed a GA for the analysis of aqueous DNA in order to generate conformations consistent with NMR distance constraints.[45] A binary encoding was used with torsional angles encoded as binary numbers and the conformation of the ribose ring was described by two binary encoded numbers, a pucker amplitude, and pseudo-rotational phase angle. To reduce the search space a hierarchical system of GAs was devised. At the bottom of the hierarchy GAs performed conformational analysis on single DNA bases. GAs were then employed higher in the hierarchy to link the DNA structure together.

3.1.3 RNA Structure Prediction

RNA structure prediction remains an intractable problem, and is further complicated by a lack of experimental structural information. Nevertheless, GAs are among the current state of the art techniques in RNA structure prediction.[46] An important area of RNA structure prediction is determining which nucleotides form stem loops. Separate groups have devised different encodings for use within a GA. Two groups, Beneditti and Morosetti[47] and van Batenburg et al.,[48] identify all possible stem loops and then use a binary encoding. If a bit is set then the corresponding stem loop is present. Gultyaev et al.[48] have extended their work to simulate the folding process.[49] Alternatively, Shapiro and Navetta[50] encode the size and start and stop positions of each loop in an individual as tuples. Given the complexity of the problem a specialist annealing mutation operator has been developed by Shapiro and Wu[51] and massively parallel hardware utilized.[50]

Once the nucleotides within stem loops have been identified, it is a nontrivial task to identify the 3D structure of the loop. In the work of Orgeta et al. the 3D conformation of RNA stem loops was modeled using a GA.[52] Each GA individual contained seven variables for each nucleotide: six of each torsion and one to describe the conformation of the sugar ring.

3.2 Protein Docking

Prediction of the binding mode of a ligand within a protein active site of known structure is of paramount importance in rational drug design[53,54] (see ***Drug Design***). There have been many attempts to solve this problem using EAs. As in

protein structure prediction, the problem is twofold: a fast search algorithm is required to evaluate many possible binding modes and appreciation of the process of molecular recognition is required to design good target functions (see ***Molecular Design and Structure-based Drug Design***).

Early attempts at docking considered both the protein and ligand to be rigid. Xiao and Williams used a GA to perform rigid body docking of deoxyguanosine into actinomycin D.[55] Three translations and three rotations were encoded as binary numbers and a simple molecular mechanics term was used in the fitness function.

Investigators have found that EAs are particularly appropriate when considering ligand flexibility. Judson et al. docked a flexible inhibitor into thermolysin using a GA.[56] Rotations around flexible bonds were encoded as binary numbers and one portion of the ligand was held fixed in the active site, where a known interaction occurred. The fitness function first tested for bad contacts and then (if no bad contacts were found) a molecular mechanics energy was calculated. This method has since been verified on a series of small molecules docked into three proteins.[57] Clark and Ajay utilized a similar scheme, though one portion of the ligand was no longer constrained to interact with the protein.[58] Three rigid body translations, three rigid body rotations, and rotations around flexible bonds were encoded as binary numbers. Good results were obtained across four different complexes. The fitness function evaluated a molecular mechanics energy for the complex. Using the same encoding Verkhivker et al. have performed docking studies on HIV-protease and FK506BP.[59] The fitness function calculated simple pairwise atomic interactions. Atoms were characterized as hydrogen bond donors, acceptors, donor–acceptors, or nonpolar. Another approach is that described by G. Jones et al.[60] Here the encoding used by the GA comprised both integer and binary strings. The binary numbers encode rotations about flexible bonds and the integer strings encode possible intermolecular hydrogen bonds. Least-squares fitting was used to dock the ligand so as to try to form the encoded hydrogen bonds. The fitness function comprised a term for steric interactions and a specialized term for hydrogen bonding. Recently, the algorithm has been improved considerably and tested on 100 protein–ligand complexes.[61] Oshiro et al. describe the inclusion of a GA with the DOCK suite of programs to perform flexible ligand docking.[62,63] The encoding specifies rotations about flexible bonds and a putative mapping between protein and ligand spheres (the spheres are shape descriptors used by the DOCK program).[62] Again, a molecular mechanics energy is calculated in the fitness function.

All of the docking algorithms described above utilize GAs. Gehlhaar et al. describe a rapid docking procedure that uses an EP and its application to HIV protease.[19] An encoding of three rigid body translations, three rigid body rotations, and rotations around flexible bonds was used. The fitness function was similar to that used by Verkhivker et al.[59] and utilized simple pairwise atomic potentials.

The methods described above use specialized objective functions in order to predict ligand binding modes, though the same docking methods can be applied to more constrained docking problems. Meadows and Hajduk have devised a GA that docks ensembles of small ligands subject to NMR constraints.[64]

3.3 Small Molecule Conformational Search and Structure Prediction

Many of the techniques described previously for structure prediction of macromolecules and docking predictions can be applied to the conformation analysis of small molecules with a view to energy minimization or constraint satisfaction. (See ***Conformational Search: Linear Chains; Coarse-Grained Searches Over Protein Conformations; Conformational Analysis; Conformational Analysis: 1; Conformational Analysis: 2***; and ***Conformational Search for Medium-Sized Molecules***.)

Brodmeier and Pretsch have used a GA to perform molecular minimization using a molecular mechanics energy function.[65] The GA encodes torsional angles as binary numbers in each individual. Hermann and Suhai use a similar encoding, but use the MOPAC AM1 *Hamiltonian*[66] in their fitness function.[67] Using the same encoding Meza and co-workers found that GAs and a direct search method were equally effective and efficient at energy minimization.[68] This work was a continuation of an earlier study on model 2D systems.[69]

A common problem is the identification of molecular conformations that satisfy NMR constraints. Sanderson et al. describe the use of a GA to determine possible solution conformations of a cyclic Arg-Gly-Asp peptide analog that were consistent with NMR data.[70] Torsions were encoded as binary numbers. In order to perform full cyclic conformational search the macrocycle was broken and constraints were applied to ensure ring closure. Beckers et al. describe a distributed GA that is used to obtain biopolymer conformations consistent with NMR data.[71]

A different GA application is described by Perlman.[72] Here possible conformations that satisfy NMR constraints are determined by molecular dynamics. A GA then determines the weighted set of conformations that best satisfy the NMR data. Molecular conformations consistent with NMR data are often generated using distance geometry. van Kampen et al. have developed a GA that optimizes matrix embedding for distance geometry by encoding putative distance geometry bounds.[73]

3.4 Molecular Design

Designing new molecules possessing desired qualities is an important activity in chemical and pharmaceutical research (see ***Design of Compounds for Physical Properties***). Computer-aided molecular design offers an attractive alternative to traditional methodologies. Again EAs have proved promising in this field. Venkatasubramanian et al. have developed a GA for molecular design.[74–76] Molecules are encoded as strings using a symbolic chemical group alphabet and then optimized to have desired physical properties. Glen and Payne have devised a GA for the generation of molecules within constraints.[77] GA individuals encode 3D structure and crossover entails exchange of substructure between parent structures. Mutation may have a number of effects, such as moving the structure, changing molecular conformation, or altering the chemical composition of the structure. A variety of constraints are available, including scalar properties, surface properties, and grid properties. Westhead et al. use a GA for structure refinement within a de novo design program.[78] Again the encoding used is the chemical structures themselves.

The starting population consists of a number of compounds produced by a *de novo* design program. These are evolved to satisfy the constraints of a design model. An alternative approach to designing molecules with specific properties is given by Devillers, who uses a communicating hybrid system of a GA and neural network.[79]

De novo protein design can be viewed as a reverse folding problem, where an amino acid sequence is designed to fold to a given 3D structure. Using a GA, D. T. Jones has attempted a solution to this problem.[80] To encode sequences the GA used a symbolic alphabet of amino acid types. Using this technique Jones et al. designed a helical protein, which, on synthesis, exhibited evidence of significant helical content.[81] Schneider et al. propose the use of an ES for protein design.[16] Unfortunately as yet, the method has only been applied to a model multimodal function, though an encoding scheme based on amino acid types and conformational preferences has since been published.[82]

Desjarlais and Handel have used a GA to design the hydrophobic core of proteins.[83] They encoded the problem using a symbolic alphabet that specified residue types and residue conformations (which were picked from a rotamer library). The fitness function evaluated the steric energy of the encoded core. The algorithm was able to reproduce known core sequences and two variants of phage 434 cro protein, designed by the program, proved as stable as the native structure.

3.5 Chemical Structure Handling

One of the first applications of genetic algorithms in the field of chemical structure handling was by Fontain.[84] In order to determine the minimum chemical distance between two structures a GA was used to map one structure onto another. Integer strings were used by the GA to encode putative mappings. A common problem in the handling of chemical structures is the retrieval of structures having particular properties from large databases. Using a similar encoding to Fontain, Brown et al. describe how a GA can be used to match 2D chemical structures.[85] While the algorithm they developed was found to be considerably inferior to conventional substructure search methods, it was found to be useful in the construction of hyperstructures (where a hyperstructure is a novel representation of a set of chemical structures).[85,86] The GA was found to be considerably more effective in matching flexible 3D molecules to pharmacophores (where a pharmacophore is a 3D arrangement of atoms or features that are considered necessary for activity).[87] Torsion angles were encoded as binary numbers and the method was found to be superior to distance geometry, systematic search, and random search. However, the directed tweak torsional minimization method appeared to be slightly superior to GA search. GAs have also been used to determine the maximum overlap of molecular electrostatic potential between two rigid molecules.[88] The technique was applied to database similarity search and has been extended to incorporate molecular flexibility.[89]

An unusual application in this field is described by Le Bret, where a GA was used to reconstruct possible connectivity matrices from lists of diatomic fragments.[90] Hibbert has described how a GA can evolve a drawing of a 2D chemical structure from its molecular formula.[91]

3.6 Combinatorial Libraries and Library Synthesis

In view of the inherent complexity of combinatorial libraries, it is not surprising to find a number of EA applications in the field (see ***Combinatorial Libraries: Structure–Activity Analysis***). Sheridan and Kearsley describe how a GA can be used to suggest combinatorial library monomers.[92] Each GA individual encodes a possible library compound as a series of monomer bases. The similarity of the encoded compounds to known actives is used as a fitness function. At the end of a GA run good library compounds that resemble known actives are found.

A number of investigators have used assay results in EA fitness functions to synthesize high-activity compounds. Weber et al. encoded starting reagents for an Ugi reaction in a binary string GA.[93] The fitness function involved synthesizing the products and then measuring their inhibitory ability. Using a thrombin assay, micromolar inhibitors were found. Singh et al. used a binary string GA to encode hexapeptide sequences.[94] The fitness function used a stromelysin and collagenase activity assay to identify stromelysin-selective substrates. Using a similar GA, Yokobayashi et al. evolved trypsin-inhibiting hexapeptides.[95]

3.7 Atomic Clusters

Another combinatorial problem that has seen the successful application of EAs is the prediction of atomic clusters. Mestres and Scuseria encode topological descriptors in their GA and, after parametrization on a C_8 cluster, the algorithm is applied to rare gas atomic clusters of up to 13 atoms.[96] Gregurick et al. use a hybrid GA to perform optimization of $(Ar)_n$ and $B(Ar)_n$ clusters.[97] A binary encoding of internal atom–atom distances is used and a local minimization procedure is applied to new individuals. The idea has been extended by Niesse and Mayne, who obtain better convergence using a real-valued encoding based on Cartesian coordinates clusters.[98] Deaven and Ho have described a GA that efficiently finds fullerene clusters of up to C_{60}.[99] The cluster structure is used as the GA encoding and crossover is achieved via a cut and splice operation on the parent structures. The technique has since been extended to optimize Lennard-Jones clusters.[100] Tomasulo and Ramakrishna have used density function theory within the fitness function of a GA to reproduce the global minimum of aluminum phosphate structures.[101]

3.8 QSAR

A *QSAR* model is normally a linear combination of basis functions of molecular descriptors (such as LogP, molecular dipole moment, and so on). In building a QSAR model, the selection of features, basis functions, and basis coefficients is a complex and time-consuming task. It is not surprising to find that many workers have developed EAs to improve on conventional methods. In the genetic function approximation (GFA) developed by Rogers and Hopfinger, a GA manipulates strings of basis functions.[102] QSAR activity models are then produced by generating basis set coefficients by linear regression. By means of a similar encoding Luke has used an EP to generate QSAR models.[18] It is not clear which of the two approaches is more successful. Leardi has also performed basis selection using a GA.[103] Kubinyi has developed an EA

that is capable of basis function selection.[104] The EA is similar to EP in that it is driven by mutation, but it only maintains a population of size one. The individual comprises a set of basis functions that are used to construct a putative QSAR. In a single generation repeated mutations are attempted, and the best individual is selected for the next generation. If no improvement is found a local search is attempted. Kubinyi has also found that basis function selection can be achieved systematically for the first three basis functions.[105] Subsequent basis functions in the QSAR can then be generated using the EA. More recently the same algorithm has been applied to variable selection in regression and partial least squares (PLS) analysis.[106]

Neural networks are commonly used for activity prediction in QSAR. So and Karplus have used a genetic algorithm for selection of molecular descriptors. These are then used as input to a neural net which predicts activity.[107] An alternative approach is suggested by Kyngäs and Valjakka, who encode the neural net size and descriptors within a GA.[108] The performance of the encoded network is then used as the GA fitness function.

3.9 Crystallography

(See ***Crystal Structure Calculations: 1*** and ***Crystal Structure Calculations: 2***.)

There have been a number of diverse applications of EAs within crystallography reported in the literature. Heavy-atom replacement is a common procedure for phase assignment in protein crystallography. Chang and Lewis use a GA to search for heavy-atom sites that are consistent with the observed Patterson function difference.[109] The coordinates of putative heavy-atom sites were encoded as binary numbers on the GA string. Another application is detailed by Tam and Compton, who successfully index crystal faces using a GA.[110] Face indices were encoded in a binary lookup table and the best results were obtained using two GA runs, with the first restricted to indexing only three faces. By encoding atomic coordinates as binary numbers within a GA, Bush et al. have predicted the crystal structure of Li_3RuO_4.[111] Experimental X-ray powder diffraction profiles are in accordance with their model predictions. Finally, Miller et al. have used a GA for *de novo* phasing of low-resolution X-ray data from crystals of icosahedral viruses,[112] where bits were used to encode electron density on a lattice.

3.10 Pharmacophore and Receptor Models

In the absence of the 3D structure of a protein target, pharmacophore modeling provides a useful alternative. Unfortunately, elucidation of the pharmacophore from a series of actives is a complicated problem, especially when molecular flexibility is a consideration. Payne and Glen applied a binary-coded genetic algorithm to the problems of molecular similarity and pharmacophore elucidation.[113] The GA encoded torsions as bytes on the binary string and cyclic flexibility was accounted for by ring corner flipping. Two cases were investigated. First, a series of molecules was fitted to a known pharmacophore. Second, given only pharmacophore features, the GA was able to elucidate pharmacophore distances. The best way of achieving this was to use the GA to encode molecular conformation; an overlay of molecules was created using least-squares fitting on the pharmacophore features. The GA was then used to minimize the distance difference between pharmacophore features. An even more problematic case is when neither pharmacophore features nor distances are known. G. Jones et al. have devised a GA that, without any prior knowledge, is able to perform flexible molecular overlay and pharmacophore elucidation.[114] The GA encodes conformational information in bit strings and encodes putative pharmacophores as integer string mappings between molecules in the overlay. The fitness function is devised to guide the evolution of good pharmacophores.

Receptor modeling is a more ambitious approach to pharmacophore modeling. In this technique a model of the active site is constructed from known actives. Walters and Hinds have used a GA to attempt this task.[115] The program requires a set of pre-aligned active molecules. The GA encoding is a string of atom types. A set of possible protein atom positions was created around the aligned molecules. Atom types for the positions are obtained by decoding a GA individual. The fitness function is designed so that the evolved model has good protein–ligand interactions. The model can then be used to predict the activity of new ligands.

3.11 Quantum Mechanics

There have been a number of instances where EAs have been used to obtain approximate solutions to the Schrödinger equation. Zeiri et al. use a real-value encoding to aid in the calculation of bound states in a double well potential and in the non-linear density functional calculation.[116] Rossi and Truhlar have devised a GA to fit a set of energy differences obtained by NDDO semiempirical *Molecular Orbital Theory* to reference *ab initio* data in order to yield specific reaction parameters.[117] The technique was applied to the reaction $Cl + CH_4$. In a third example, Rodriguez et al. apply a GA to diagonalization of the Hamiltonian in the spin-coupled Fe^{3+}–Fe^{2+} center of reduced uteroferrin.[118]

3.12 Other Miscellaneous Examples

In the field of protein similarity May and Johnson have devised a GA for protein structure comparison.[119,120] Their binary string GA encodes three translations and three rigid body rotations. On decoding an individual a transformation is generated to superimpose one protein on top of another. The fitness function uses a dynamic programming technique to determine topological equivalences.

Jaeger et al. have reported using a GA to design a primary screen for antirhinovirus agents. The activity of a diverse set of human rhinovirus inhibitors was measured against a larger set of rhinovirus serotypes and a GA then selected serotypes for the new screen.[121]

The use of GAs in partial least-squares (PLS) regression has been the subject of a few publications (see ***Partial Least Squares (PLS) in Chemistry***). Shaffer et al. have used a GA to guide the coupling of bandpass digital filtering and PLS in the near-infrared analysis of glucose in biological matrices.[122] Bangalore et al. used a GA to complement an automated wavelength selection procedure for use in building multivariate calibration models based on PLS.[123] GAs have also been used by

Vankeerberghen et al. to obtain parameters for robust regression in nonlinear models, which can then be used for instrument calibration.[124] Wise et al. have found GAs to be more efficient than neural nets in developing non-linear models.[125]

There are many other reported applications of GAs in related fields. Some, which may be of interest, include: finding low-energy distributions of ions above a crystal surface;[126] optimizing parameters for allosteric regulation of enzymes in a model metabolic cycle;[127] minimizing the discrepancies between experimental and theoretical fluorescence/absorption spectra;[128] thin film analysis;[129] and parameter estimation for kenetic models.[130,131]

4 CONCLUSIONS

It is only since the early 1990s that EAs have been applied to problems in computational chemistry. Yet, in that short space of time, they have had remarkable success in almost every complex search problem within the field. This success must be due to the problem independence of EAs. Given a problem, all that is required to implement an EA is appropriate encoding and fitness functions. It is hoped that the many applications discussed here are sufficient to demonstrate the potential of this powerful new tool for the investigation of a wide range of optimization problems in chemistry and biology.

Given this initial success, it would seem probable to expect, in the future, an increasing number of powerful and sophisticated EA applications applied to complex problems in the field of computational chemistry.

5 RELATED ARTICLES

Artificial Intelligence in Chemistry; Conformational Search: Linear Chains; Chemometrics: Multivariate View on Chemical Problems; Coarse-Grained Searches Over Protein Conformations; Combinatorial Libraries: Structure–Activity Analysis; Conformational Analysis; Conformational Analysis: 1; Conformational Analysis: 2; Conformational Search for Medium-Sized Molecules; Crystal Structure Calculations: 1; Design of Compounds for Physical Properties; Drug Design; Fuzzy Methods in Chemistry; Genetic Algorithms: Introduction and Applications; Molecular Design and Structure-based Drug Design; Crystal Structure Calculations: 2; Partial Least Squares (PLS) in Chemistry; Protein Folding and Optimization Algorithms; Protein Structure Prediction in 1D, 2D, and 3D.

6 REFERENCES

1. C. Darwin, 'The Origin of Species', Dent Gordon, London, 1973.
2. T. Bäck and H.-P. Schwefel, *Evol. Comput.*, 1993, **1**, 1–23.
3. T. Bäck, 'Evolutionary Algorithms in Theory and Practice', Oxford University Press, Oxford, 1996.
4. D. B. Fogel, 'Evolutionary Computation: Toward a New Philosophy of Machine Intelligence', IEEE Press, Piscataway, NJ, 1995.
5. J. H Holland, 'Adaptation in Natural and Artificial Systems', MIT Press, Cambridge, MA, 1992.
6. D. E. Goldberg, 'Genetic Algorithms in Search Optimization and Machine Learning', Addison-Wesley, Reading, MA, 1989.
7. I. Rechenberg, 'Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution', Frommann-Holzboog, Stuttgart, 1973.
8. H.-P. Schwefel, 'Numerical Optimization of Computer Models', Wiley, Chichester, 1981.
9. L. J. Fogel, A. J. Owens, and M. J. Walsh, 'Artificial Intelligence through Simulated Evolution', Wiley, New York, 1966.
10. J. Antonisse, in 'Proceedings of the Third International Conference on Genetic Algorithms', ed. D. Schaffer, Morgan Kaufmann, San Mateo, CA, 1989, pp. 86–91.
11. L. Davis, (ed.), 'Handbook of Genetic Algorithms', Van Nostrand Reinhold, New York, 1991.
12. I. M. Oliver, D. J. Smith, and J. R. C. Holland, in 'Proceedings of the Second International Conference on Genetic Algorithms', ed. J. J. Grefensette, Lawrence Erlbaum Associates, London, 1987, pp. 224–230.
13. Z. Michalewicz, 'Genetic Algorithms + Data Structures = Evolution Programs', Springer, Berlin, 1992.
14. N. J. Radcliffe, in 'Proceedings of the Fourth International Conference on Genetic Algorithms', eds. R. K. Belew and L. B. Booker, Morgan Kaufmann, San Mateo, CA, 1991, pp. 222–229.
15. D. Whitley, in 'Proceedings of the Third International Conference on Genetic Algorithms', ed. D. Schaffer, Morgan Kaufmann, San Mateo, CA, 1989, pp. 116–121.
16. G. Schneider, J. Schuchhardt, and P. Wrede, *Comput. Appl. Biosci.*, 1994, **6**, 635–645.
17. D. B. Fogel, 'Evolving Artificial Intelligence', Ph.D. Thesis, University of California, San Diego, CA, 1992.
18. B. T. Luke, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 1279–1287.
19. D. K. Gehlhaar, G. M. Verkhivker, P. A. Rejto, C. J. Sherman, D. B. Fogel, L. J. Fogel, and S. T. Freer, *Chem. Biol.*, 1995, **2**, 317–324.
20. D. E. Clark and D. R. Westhead, *J. Comput.-Aided Mol. Des.*, 1996, **10**, 337–358.
21. P. Willett, *Trends Biotechnol.*, 1995, **13**, 516–521.
22. A. L. Parrill, *Drug Discovery Today*, 1996, **1**, 514–521.
23. J. T. Pedersen and J. Moult, *Curr. Opin. Struct. Biol.*, 1996, **6**, 227–231.
24. P. Judson, in 'Reviews in Computational Chemistry', Vol. 10, eds. K. B. Lipkowitz and D. B. Boyd, Wiley, Chichester, 1997, pp. 1–73.
25. J. Devillers, 'Genetic Algorithms in Molecular Modelling', Academic Press, London, 1996.
26. R. Unger and J. Moult, *J. Mol. Biol.*, 1993, **231**, 75–81.
27. J. U. Bowie and D. Eisenberg, *Proc. Natl. Acad. Sci. USA*, 1994, **91**, 4436–4440.
28. T. Dandekar and P. Argos, *Protein Eng.*, 1992, **5**, 637–645.
29. T. Dandekar and P. Argos, *J. Mol. Biol.*, 1994, **236**, 844–861.
30. T. Dandekar and P. Argos, *Int. J. Biol. Macromol.*, 1996, **18**, 1–4.
31. T. Dandekar and P. Argos, *J. Mol. Biol.*, 1996, **256**, 645–660.
32. S. Sun, *Protein Sci.*, 1993, **2**, 762–785.
33. S. Sun, P. D. Thomas, and K. A. Dill, *Protein Eng.*, 1995, **8**, 769–778.
34. S. Sun, *Biophys. J.*, 1995, **69**, 340–355.
35. S. M. Le Grand and K. M. Merz Jr, *Mol. Simul.*, 1994, **13**, 299–320.
36. J. T. Pedersen and J. Moult, *Proteins: Struct. Funct. Genet.*, 1995, **23**, 454–460.
37. A. A. Rabow and H. A. Scheraga, *Protein Sci.*, 1996, **5**, 1800–1815.
38. A. Elofsson, S. M. Le Grand, and D. Eisenberg, *Proteins: Struct. Funct. Genet.*, 1995, **23**, 73–82.
39. J. R. Gunn, A. Monge, R. A. Friesner, and C. H. Marshall, *J. Phys. Chem.*, 1994, **98**, 702–711.
40. C. A. Del Carpio, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 258–269.
41. C. S. Ring and F. E. Cohen, *Isr. J. Chem.*, 1994, **34**, 245–252.

42. P. Tufféry, C. Etchebest, S. Hazout, and R. Lavery, *J. Biomol. Struct. Dyn.*, 1991, **6**, 1267–1287.
43. P. Tufféry, C. Etchebest, S. Hazout, and R. Lavery, *J. Comput. Chem.*, 1993, **14**, 790–798.
44. F. Vivarelli, G. Giusti, M. V. R. Campanini, M. Compiani, and R. Casadio, *Comput. Appl. Biosci.*, 1995, **11**, 253–260.
45. C. B. Lucasius, M. J. J. Blommers, L. M. C. Buydens, and G. Kateman, in 'Handbook of Genetic Algorithms', ed. L. Davis, Van Nostrand Reinhold, New York, 1991, pp. 251–281.
46. S. Louise-May, P. Auffinger, and E. Westhof, *Curr. Opin. Struct. Biol.*, 1996, **6**, 289–298.
47. G. Beneditti and S. Morosetti, *Biophys. Chem.*, 1995, **55**, 253–259.
48. F. H. D. van Batenburg, A. P. Gultyaev, and W. A. Pleu, *J. Theor. Biol.*, 1995, **174**, 269–280.
49. A. P. Gultyaev, F. H. D. van Batenburg, and C. W. A. Pleij, *J. Mol. Biol.*, 1995, **250**, 37–51.
50. B. A. Shapiro and J. J. Navetta, *J. Supercomputing*, 1994, **8**, 195–207.
51. B. A. Shapiro and J. C. Wu, *Comput. Appl. Biosci.*, 1996, **12**, 171–180.
52. H. Ogata, Y. Akiyama, and M. Kanehisa, *Nucleic Acids Res.*, 1995, **23**, 419–426.
53. G. Jones and P. Willett, *Curr. Opin. Biotechnol.*, 1995, **6**, 652–656.
54. T. P. Lybrand, *Curr. Opin. Struct. Biol.*, 1995, **5**, 224–228.
55. Y. L. Xiao and D. E. Williams, *J. Phys. Chem.*, 1994, **98**, 7191–7200.
56. R. S. Judson, E. P. Jaeger, and A. M. Treasurywala, *J. Mol. Struct. (Theochem)*, 1994, **308**, 191–206.
57. R. S. Judson, Y. T. Tan, E. Mori, C. Melius, E. P. Jaeger, A. M. Treasurywala, and A. J. Mathioweitz, *J. Comput. Chem.*, 1995, **16**, 1405–1419.
58. K. P. Clark and Ajay, *J. Comput. Chem.*, 1995, **16**, 1210–1226.
59. G. M. Verkhivker, P. A. Rejto, D. K. Gehlhaar, and S. T. Freer, *Proteins: Struct. Funct. Genet.*, 1996, **250**, 342–353.
60. G. Jones, P. Willett, and R. C. Glen, *J. Mol. Biol.*, 1995, **245**, 43–53.
61. G. Jones, P. Willett, R. C. Glen, A. R. L. Leach, and R. Taylor, *J. Mol. Biol.*, 1997, **267**, 727–748.
62. I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin, *J. Mol. Biol.*, 1982, **161**, 269–288.
63. C. M. Oshiro, I. D. Kuntz, and J. S. Dixon, *J. Comput.-Aided Mol. Des.*, 1995, **9**, 113–130.
64. R. P. Meadows and P. J. Hajduk, *J. Biomol. NMR*, 1995, **5**, 41–47.
65. T. Brodmeier and E. Pretsch, *J. Comput. Chem.*, 1994, **15**, 588–595.
66. J. P. P. Stewart, *J. Comput.-Aided Mol. Des.*, 1990, **4**, 1–45.
67. F. Hermann and S. Suhai, *J. Comput. Chem.*, 1995, **16**, 1434–1444.
68. J. C. Meza, R. S. Judson, T. R. Faulkner, and A. M. Treasurywala, *J. Comput. Chem.*, 1996, **17**, 1142–1151.
69. J. C. Meza and M. L. Martinez, *J. Comput. Chem.*, 1994, **15**, 627–632.
70. P. N. Sanderson, R. C. Glen, A. W. R. Payne, B. D. Hudson, C. Heide, G. E. Tranter, P. M. Doyle, and C. J. Harris, *Int. J. Pept. Protein Res.*, 1994, **43**, 588–596.
71. M. L. M. Beckers, E. P. P. A. Derks, W. J. Melsen, and L. M. C. Buydens, *Comput. Chem. Eng.*, 1996, **20**, 449–457.
72. D. A. Perlman, *J. Biomol. NMR*, 1996, **8**, 49–66.
73. A. H. C. van Kampen, L. M. C. Buydens, C. B. Lucasius, and M. J. J. Blommers, *J. Biomol. NMR*, 1996, **7**, 214–224.
74. V. Venkatasubramanian, K. Chan, and J. M. Caruthers, *Comput. Chem. Eng.*, 1994, **18**, 833–844.
75. V. Venkatasubramanian, K. Chan, and J. M. Caruthers, *ACS Symp. Ser.*, 1995, **589**, 396–411.
76. V. Venkatasubramanian, K. Chan, and J. M. Caruthers, *J. Chem. Inf. Comput. Sci.*, 1995, **35**, 188–195.
77. R. C. Glen and A. W. R. Payne, *J. Comput.-Aided Mol. Des.*, 1995, **9**, 181–202.
78. D. R. Westhead, D. E. Clark, D. Frenkel, J. Li, C. W. Murray, B. Robson, and B. Waszkowycz, *J. Comput.-Aided Mol. Des.*, 1995, **9**, 139–148.
79. J. Devillers, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 1061–1066.
80. D. T. Jones, *Protein Sci.*, 1994, **3**, 567–574.
81. D. T. Jones, C. M. Moody, J. Uppenbrink, J. H. Viles, P. M. Doyle, J. C. Harris, L. H. Pearl, P. J. Sadler, and J. M. Thornton, *Proteins: Struct. Funct. Genet.*, 1996, **24**, 502–513.
82. G. Schneider, J. Schuchhardt, and P. Wrede, *Endocytobiosis Cell Res.*, 1995, **11**, 1–18.
83. J. R. Desjarlais and T. M. Handel, *Protein Sci.*, 1995, **4**, 2006–2018.
84. E. Fontain, *J. Chem. Inf. Comput. Sci.*, 1992, **32**, 748–752.
85. R. D. Brown, G. Jones, and P. Willett, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 63–70.
86. R. D. Brown, G. M. Downs, G. Jones, and P. Willett, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 47–53.
87. D. E. Clark, G. Jones, and P. Willett, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 197–206.
88. D. J. Wild and P. Willett, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 159–167.
89. D. A. Thorner, D. J. Wild, P. Willett, and P. M. Wright, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 900–908.
90. C. Le Bret, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 678–683.
91. D. B. Hibbert, *Chemometr. Intell. Lab. Syst.*, 1993, **20**, 35–43.
92. R. P. Sheridan and S. K. Kearsley, *J. Chem. Inf. Comput. Sci.*, 1995, **35**, 310–320.
93. L. Weber, S. Wallbaum, C. Broger, and K. Gubernator, *Angew. Chem., Int. Ed. Engl.*, 1995, **34**, 2280–2282.
94. J. Singh, M. A. Ator, E. P. Jaeger, M. P. Allen, D. A. Wipple, J. E. Soloweij, S. Chowdhary, and A. M. Treasurywala, *J. Am. Chem. Soc.*, 1996, **118**, 1669–1676.
95. Y. Yokobayashi, K. Ikebukuro, S. McNiven, and I. Karube, *J. Chem. Soc., Perkin Trans. 1*, 1996, 2435–2437.
96. J. Mestres and G. E. Scuseria, *J. Comput. Chem.*, 1995, **16**, 729–742.
97. S. K. Gregurick, M. H. Alexander, and B. Hartke, *J. Chem. Phys.*, 1996, **104**, 2684–2691.
98. J. A. Niesse and H. R. Mayne, *J. Chem. Phys.*, 1996, **105**, 4700–4706.
99. D. M. Deaven and K. M. Ho, *Phys. Rev. Lett.*, 1995, **75**, 288–291.
100. D. M. Deaven, N. Tit, J. R. Morris, and K. M. Ho, *Chem. Phys. Lett.*, 1996, **256**, 195–200.
101. A. Tomasulo and M. V. Ramakrishna, *J. Phys. Chem.*, 1996, **105**, 10449–10455.
102. D. Rogers and A. J. Hopfinger, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 854–866.
103. R. Leardi, *J. Chemometr.*, 1994, **8**, 65–79.
104. H. Kubinyi, *Quant. Struct.-Act. Relat.*, 1994, **13**, 285–294.
105. H. Kubinyi, *Quant. Struct.-Act. Relat.*, 1994, **13**, 393–401.
106. H. Kubinyi, *J. Chemometr.*, 1996, **10**, 119–133.
107. S. S. So and M. Karplus, *J. Med. Chem.*, 1996, **39**, 1521–1530.
108. J. Kyngäs and J. Valjakka, *Quant. Struct.-Act. Relat.*, 1996, **15**, 296–301.
109. G. Chang and M. Lewis, *Acta Crystallogr.*, 1994, **D50**, 667–674.
110. K. Y. Tam and R. G. Compton, *J. Appl. Crystallogr.*, 1995, **28**, 640–645.
111. T. S. Bush, C. R. A. Catlow, and P. D. Battle, *J. Mater. Chem.*, 1995, **5**, 1269–1272.
112. S. T. Miller, J. M. Hogle, and D. J. Filman, *Acta Crystallogr.*, 1996, **D52**, 235–251.

113. A. W. R. Payne and R. C. Glen, *J. Mol. Graphics*, 1993, **11**, 74–91.
114. G. Jones, P. Willett, and R. C. Glen, *J. Comput.-Aided Mol. Des.*, 1995, **9**, 532–549.
115. D. E. Walters and R. M. Hinds, *J. Med. Chem.*, 1994, **37**, 2527–2536.
116. Y. Zeiri, E. Fattal, and R. Kosloff, *J. Chem. Phys.*, 102, **4**, 1859–1862.
117. I. Rossi and D. G. Truhlar, *Chem. Phys. Lett.*, 1995, **233**, 231–236.
118. J. H. Rodriguez, H. N. Ok, Y.-M Xia, P. G. Debrunner, B. E. Hinrichs, T. Meyer, and N. H. Packard, *J. Phys. Chem.*, 1996, **100**, 6849–6862.
119. A. C. W. May and M. S. Johnson, *Protein Eng.*, 1994, **7**, 475–485.
120. A. C. W. May and M. S. Johnson, *Protein Eng.*, 1995, **8**, 873–882.
121. E. P. Jaeger, D. C. Pevear, P. J. Felock, G. R. Russo, and A. M. Treasurywala, *ACS Symp. Ser.*, 1995, **589**, 139–155.
122. R. E. Shaffer, G. W. Small, and M. A. Arnold, *Anal. Chem.*, 1996, **68**, 2663–2675.
123. A. S. Bangalore, R. E. Shaffer, G. W. Small, and M. A. Arnold, *Anal. Chem.*, 1996, **68**, 4200–4212.
124. P. Vankeerberghen, J. Smeyers-Verbeke, R. Leardi, C. L. Karr, and D. L. Massart, *Chemometr. Intell. Lab. Syst.*, 1995, **28**, 73–●●●.
125. B. M. Wise, B. R. Holt, N. B. Gallager, and S. Lee, *Chemometr. Intell. Lab. Syst.*, 1995, **30**, 81–89.
126. A. H. C. van Kampen, C. S. Strom, and L. M. C. Buydens, *Chemometr. Intell. Lab. Syst.*, 1996, **34**, 55–68.
127. A. Gilman and G. Ross, *Biophys. J.*, 1995, **69**, 1321–1333.
128. J. Dods, D. Gruner, and P. Brumer, *Chem. Phys. Lett.*, 1996, **261**, 612–619.
129. A. D. Dane, P. A. M. Timmermans, H. A. van Sprang, and L. M. C. Buydens, *Anal. Chem.*, 1996, **68**, 2419–2425.
130. D. B. Hibbert, *Chemometr. Intell. Lab. Syst.*, 1993, **19**, 319–329.
131. R. Moros, H. Kalies, H. G. Rex, and S. Schaffarczyl, *Comput. Chem. Eng.*, 1996, **20**, 1257–1270.