Evolutionary-Algorithm-Based Strategy for Computer-Assisted Structure Elucidation

Yongquan Han and Christoph Steinbeck*,†

Max-Planck-Institut für Chemische Ökologie, Hans-Knöll-Strasse 8, 07745 Jena, Germany

Received July 1, 2003

An evolutionary algorithm (EA) using a graph-based data structure to explore the molecular constitution space is presented. The EA implementation proves to be a promising alternative to deterministic approaches to the problem of computer-assisted structure elucidation (CASE). While not relying on any external database, the EA-guided CASE program SENECA is able to find correct solutions within calculation times comparable to that of other CASE expert systems. The implementation presented here significantly expands the size limit of constitutional optimization problems treatable with evolutionary algorithms by introducing novel efficient graph-based genetic operators. The new EA-based search strategy is discussed including the underlying data structures, component design, parameter optimization, and evolution process control. Typical structure elucidation examples are given to demonstrate the algorithm's performance.

1. INTRODUCTION

Structure elucidation of an unknown compound, based on knowledge of its molecular formula, spectral data, and other prior information, is a process of searching the best-matched constitutional formula among usually isomeric structure candidates. The constitutional isomers for a given molecular formula constitute the so-called constitution space—an assembly of finite-numbered isomers. How these isomers are distributed in such a discrete space is not unveiled. To the best of the authors' knowledge, there is no practical measure available. Due to the implicit existence of constitutional constraints, an indiscriminate tiny substitution in one structure is liable to result in an ill-formed structure. The amount of the isomers is usually large, and the size of the constitution space expands exponentially proportional to the number of skeletal atoms in the molecule of the unknown. The complexity of the constitution space makes its exploration difficult and appeals elaborate strategies.

Two strategies for searching constitution spaces—deterministic as well as stochastic approaches—have been described in the literature.[1−4] A deterministic approach exhaustively searches the problem space, thus guaranteeing the optimum to be found if the algorithm has enough time to finish its job and if the constraints are error-free. Deterministic searches have been successfully applied for structure elucidation of small molecules, and thus the predominant computer-assisted structure elucidation (CASE) programs available are driven by deterministic search engines. The major drawback of this method is its deadly demand for computing power in the case of large molecules due to combinatorial explosion. Much effort has thus been devoted to dwindling the search space before the real generation and evaluation step, by deducing more structural constraints from the spectra data with the help of previous knowledge accumulations.[1,5−7] Libraries of structural fragments connected to certain spectral features are a typical way to dramatically decrease the size of the search space by combining nodes from the atom set into larger fragments, with those atoms thus no longer taking part in the combinatorial process. In the case of proton-rich compounds, large numbers of constraints derived from 2D NMR long-range correlations also cut down the search space significantly.

Our group is interested in applying the stochastic approaches to problems of CASE. In contrast to deterministic methods, a stochastic approach runs a randomized but guided search in constitution space. Starting with one or more initial structures, it evolves candidates into ones with more desired properties. A stochastic search may locate its target quickly, as it bypasses the combinatorial explosion by focusing on the most profitable (and remarkably small) regions in constitution space while not entirely neglecting the others. And a stochastic method relies little on extra knowledge bases compared to efficient deterministic approaches, as discussed in greater detail below.

Due to its random nature, however, a single stochastic optimization may not guarantee the global optimum will be found at all, which is likely to be one of the reasons why, despite its conceptual advantages over the deterministic approach, the stochastic method is not generally realized as the method of first resort in CASE. The danger of not finding the optimum in a CASE problem is usually regarded as a serious problem, since the user is interested in the one, single correct structure and not just a good one that reasonably fits some given constraints. We can show, however, that at least for molecular sizes usually treated by deterministic systems, our stochastic approaches find all correct solutions within about the same order of magnitude in calculation time.

In this paper, we present a unique evolutionary algorithm (EA) for tackling the CASE problem. The new EA implementation integrates the three steps in a CASE expert system—structure inference, structure generation, and structure verification[1]—into one procedure. This approach adopts a graph-based representation of the molecular structure and

* Corresponding author phone: ++49 (0)221-470 7426; fax: ++49 (0)221-470 7786; e-mail: c.steinbeck@uni-koeln.de.

† Present address: Cologne University Bioinformatics Center, Zülpicher Strasse 47, 50674 Köln, Germany.

a suite of robust graph operators. The labeled molecular graph data structure facilitates efficient genetic manipulation and exempts from the transformation between genotype and phenotype of the candidate solution. A flexible parameter control enables the genetic operators to adjust their behavior and achieve higher search efficiency. In the following sections, the evolutionary algorithm is explained in detail, and its performance is demonstrated by experimental results.

2. EVOLUTIONARY ALGORITHMS

An EA acts as a crude version of species evolution.[8] It inherits from nature the principles such as natural selection and survival of the fittest. A typical EA starts with an initial population of candidate solutions. Each solution is evaluated by a fitness function and assigned with a value indicating its relative correctness. The population evolves over generations by applying reconstruction operators on selected solutions. The selection of solutions, which are allowed to survive from one generation to the next one, is biased to those with higher fitness values. The algorithm terminates when satisfying solutions are found. Recently, EAs have been applied for problems such as lead structure discovery and optimization and computer-assisted molecule design (see refs 9−15). Most of these applications involve the chemical structure search in a conformational space. Yet there are few examples in which EAs are adopted to explore the constitution space (see refs 4 and 16−18). These attempts, albeit designed for different purposes, were all unable to release the EA's power due to some restrictions in their design. The approaches were restrained in traditional data structures for molecule representation using coding schemes based on (bit-) strings or trees. These representations either do not cover the entire constitution space or are inappropriate for direct and efficient genetic operations. In Reference 16 is an application of GA in molecular design and circuit design. It is not equivalent to the problem of our interest due to a different definition of search space. This access is interesting here in that it also works on direct graph operation. Its reproduction operators have some limit to cover full search space; e.g. cross-linked structures might not be generated. References 4 and 18 addressed the GA approach for the CASE problem solely on the basis of ^{13}C NMR data. The genetic operators used are a traditional mutation operator and a one- or two-point crossover operator. This method did not take measure to prevent generation of ill-formed candidates by genetic operation. There is also a risk that new structures are not connected or saturated. As a result, the genetic operators impose a large computational overhead on the EA implementation, preventing it from being applied to molecules larger than, for example, 20 heavy atoms. And above all, because the approach inherit directly (and simply) from the traditional EA paradigm, it is not able to guarantee that one of the correct structures will be discovered, let alone all of them.

To apply evolutionary algorithms to CASE problems, however, there is no doubt that a system needs to be able to find the full set of correct solutions with a good probability in order to be accepted by the user community.

As could be shown in ref 3, the fitness function of a stochastic algorithm based on spectroscopic data can be constructed such that the maximum possible fitness value is known. The awareness of its target value allows an EA to detect whether it is being trapped in a local optimum. In case the search is in a local optimum, the algorithm can take measures by automatically modifying its parameters to escape the local optimum. Taking the target value as one of the stopping criteria, a carefully designed EA should thus be able to find at least one structure complying with all input constraints.

As will be shown below, the structures fully complying with the input constraints are of great chemical similarity, and therefore the optimal structures are located in a small region, a niche, in the constitution space. New correct structures, if they exist, are to be screened out by applying a niche search around each of the known correct structures. The niche search makes the hit list complete with small extra computing effort, which is affordable even for large organic compounds.

In a deterministic search based CASE system, it is often necessary to have a verification step for hit structures, through spectra prediction and comparison, so as to identify the fittest structure among the solution set. In an EA-based strategy, this step is in no need, because the EA can implicitly perform the spectrum prediction during the evolution run by employing a HOSE-code-based[19,20] NMR shift prediction as part of the fitness function. This HOSE code evaluation function is under continuous development to achieve higher discriminability for chemical environments of carbon atoms in the questioned molecule, coupled with the open-source and open-access web database NMRShiftsDB.[21]

3. ALGORITHM DESIGN AND IMPLEMENTATION

Any EA framework should at least include an appropriate representation of the plausible solution to the problem under investigation, a fitness function scoring the candidates, a set of reproduction operators prompting evolution, and strategies of initialization, selection, and termination. Figure 1 illustrates the workflow of an evolutionary algorithm framework based on a single population. The individual components of this concept will be discussed in the following.

3.1. Graph-Based Representation. The choice of the central data structure, the representation of candidate solutions, has a crucial impact on the EA performance. There are a number of ways to represent a molecular structure in a computer, such as the connectivity matrix and its variants, trees, molecular graphs, structure descriptors, and line notations, etc. The connectivity matrix and its variants are the most common coding method for molecular structures, yet its sparse structure leads to great time and space cost (computational complexity in many cases $O(n^2)$) for storage and manipulation. Line notations, like SMILES,[22,23] on the other hand, are very compressed exact representations for storing, retrieving, and communicating constitutional information, but the manipulation of linear data structures by recombination and mutation operators will inevitably lead to invalid structures violating basic chemical valence rules. Upon careful study of the pioneering works of Globus,[16] Nachbar,[17] and Meiler,[4] we concluded that direct manipulation of an object-oriented graph representation under constant control of the basic chemical valence rules should be the most efficient way to perform genetic operations on molecular constitutional structures.

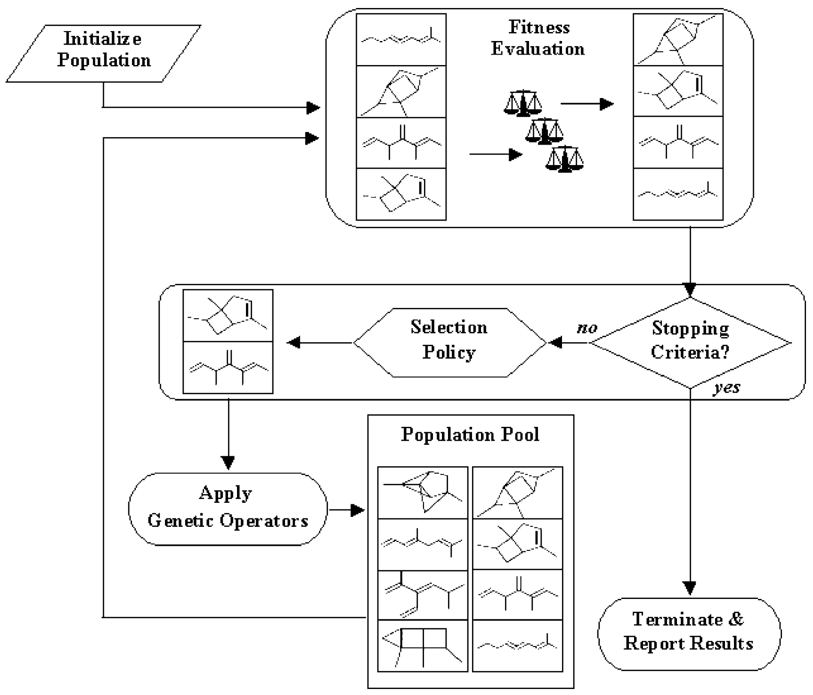


Figure 1. Evolutionary search flow chart for a single population based EA (modified from ref 28).

For the implementation described in the following, we took advantage of the chemistry development kit (CDK),[24] an open-source Java class library for structural chemo- and bioinformatics, developed by our group and a team of international collaborators.

Within the CDK, a molecular structure is coded as a set of atom objects with connectivity information stored in bond objects, all contained in a data structure called an AtomContainer. Manipulations of the AtomContainer throughout our optimization procedure keep the atom array ordered and fixed, while the bond array is varied, with the overall sum of bond orders being constant. The hydrogen atoms and hydrogen-involving bonds in the molecule are treated as implicitly belonging to certain heavy atoms—a usual procedure in constitutional cheminformatics—and are thus not taking part in the combinatorial process. It should be noted that due to this measure, one will currently have to run a separate calculation for every possible assignment of exchangeable protons to heteroatoms. In future versions of our EA engine, this will be circumvented by letting exchangeable protons take part in the combinatorial process. A variety of methods is provided for structure manipulation in the CDK. One can add or delete an atom or bond, modify bond order, split a structure in two, merge two fragments in one, and so on.

To achieve the highest possible efficiency of the evolutionary process, atoms in an AtomContainer can be tagged with particular properties, like the already mentioned HCount or the attribute atomStateTag, a Boolean variable which tells whether an atom in the AtomContainer instance is active or dormant. When an atom is in the dormant state, there is no way to change its bond type. A set of frozen atoms and the bonds between them forms a substructure free of breakage in the evolutionary process. A dormant atom may wake up triggered by a certain control parameter. This measure is desired in restricting the search to certain niches in constitutional space.

Another important aspect in order to explain the efficiency of our implementation is that each carbon atom in the AtomContainer instance has an attribute chemicalShift to save its chemical shift value. The atoms in the atom array are sorted in a descending sequence of the chemical shifts of carbon atoms, followed by other heteroatoms. This makes the molecular structure to be a *labeled* molecular graph. Carbon atoms of the same proton connections but different chemical shifts are seen as nonequivalent atoms. The traditional concept of structure isomorphism is not applicable here; a pair of differently labeled, “traditionally isomorphic” molecular graphs may have different fitness values and are thought of as being positioned in distinct points in search space.

This EA implementation is not trying to use the labeled molecular graph as the canonical one. The isomorphism problem is not solved, but got less serious here for the specific application. While the introduction of the attribute chemicalShift does not make the labeled graph a canonical one due to at least two reasons, the possible existence of structural symmetry and the randomness of the heteroatom labeling, the probability that two structures of random choice are isomorphic is decreased.

Embodying constraints into the representation, as many as possible, as early as possible, is one of the central paradigms used in the evolutionary algorithm described here. The size of constitutional space diminishes dramatically every time a new, nonredundant type of constraint is imported. For example, there are nearly 25 000 constitutional variants for the molecular formula $C_{10}H_{16}$, while when DEPT NMR spectrum information is considered, i.e., three CH, two CH_2, and three CH_3 are accepted as obligatory fragments, the number of isomer entries drops to 4306.

3.2. Genetic Operators. Corresponding to the graph-based representation, a suite of graph operators is devised to rule and propel the exploration of constitution space. Some criteria are firmly observed in the reproduction operator

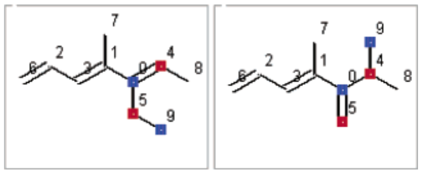


Figure 2. Mutation operation on a structure with molecular formula $C_{10}H_{16}$, with (A, left) the parent structure and (B, right) the offspring structure.

design. First, the set of used operators must be capable to construct any plausible structures in search space. Otherwise, "blind spots" arise in constitution space, with the danger of the correct structure being hidden from the algorithm in one of these blind spots.

The second consideration is to deal with violation of constraints, both chemical and others. It is possible to have a fitness function that severely penalizes ill-formed structures. This is simple but expensive, as lots of resources are spent creating and then rejecting structures that are not real candidates.[4] Especially, when the problem is of high dimensionality, almost all of the offspring structures are invalid, and so much of the processing time would be wasted.

We prefer to make the reproduction operators aware of these restrictions and ensure that each structure created can only be a valid one. This tends to be complex in design but economical in run time.

Mutation. In the simulated annealing search engine presented in an earlier publication,[3] we implemented a mutation operator suggested by Faulon.[2] For a molecular structure, the mutation operator adjusts the bond orders (with bond order 0 meaning no bond) between four arbitrary atoms while keeping the rest of the structure chemically valid. The structural validity is guaranteed by a set of valence equations. Figure 2 illustrates a mutation operation on a structure with molecular formula $C_{10}H_{16}$. In the parent structure, atoms labeled as 0, 4, 5, and 9 are selected for operation. After mutation, the bond linking atoms 5 and 9 are deleted; atom 5 connects to 0 with a double instead of a single bond; the increased bond degree of atom 0 is balanced by lowering the order of bond coupling atoms 0 and 4. A single bond is formed between atoms 4 and 9.

The mutation operator tends to do a local refinement. This process involves only four distinct atoms forming the working unit, and at most four bonds between these atoms are reshuffled. Thus the offspring has little difference from its ancestor.

The originally proposed mutation operator is able to restrict its destruction scale, but unfortunately not its action target. To fix this imperfection, we applied further restrictions to the selection of the working atoms. A parameter *mutation radius* is defined as the maximum distance (the number of bonds) from the randomly selected seed atom to other members in the working unit. The mutation radius delimits a neighborhood in which the bond type is allowed to change. The remaining part of the structure will survive to the offspring. A mutation operator becomes more disruptive with the increase of its mutation radius. For example in Figure 3, the atom marked with a black dot is selected as the seed atom and the mutation radius assigned a value of 3. With this constraint, the structural reshuffle by mutation is limited to the indicated radius, while the rest of the structure stays intact.

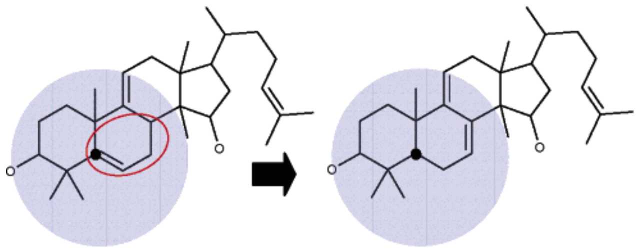


Figure 3. Effect of the mutation radius. A value of 3 is chosen in this example. The atom marked with a black dot in the parent structure was selected as the seed atom. The atoms within intervals of three intervening bonds are qualified, and four of them (within the ellipse) are selected to take part in mutation. The structural segment outside the color-filled circle survives to offspring.

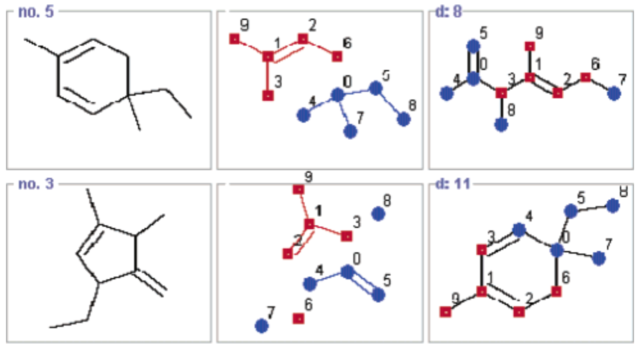


Figure 4. Crossover operation on a pair of structures with molecular formula $C_{10}H_{16}$. The minimum chemical distance between an offspring and its ancestor is given by the parameter *d*.

Crossover. In the context of constitutional optimization, crossover is the process of cleaving, merging, and saturating two parent structures to form two children, as shown in Algorithm 1. A summary of the crossover algorithm.

1. Select parent structures from population pool
2. Decide cutting scale and mode
3. Split parent structures in two fragment clusters each
4. Partition skeletal atoms into two sets
5. Preserve bond that connects two atoms in the same set, remove bond that connects two atoms from different set
6. Merge the opposite cluster from each parent and obtain a pair unsaturated offspring structures
7. Saturate offspring structures.

algorithm 1

Figure 4 shows two structures selected as parents from the population pool. The skeleton atoms of each of the parent structures are randomly segmented into two sets, square and round. For parent 1, the square set consists of atoms labeled as 1, 2, 3, 6, and 9. The round set contains atoms 0, 4, 5, 7, and 8. A similar partitioning scheme is used for parent 2. Parent 2 is split according to the atom index partitioning obtained for parent 1. For the crossover operation, bonds connecting two atoms in the same atom set are preserved, while those connecting two atoms from different sets are deleted (see bonds 3−4 and 6−0).

The two resulting fragments of each parent structure are now crosswise combined to form two offspring. The square cluster in parent 1 combines with the round cluster in parent 2, and the round cluster in parent 1 joins the square cluster in parent 2. By doing so, two incomplete structures are obtained with some atoms being unsaturated. After appending the missing bonds in the offspring structures (6−7, 0−3, and 3−8 in child structure 1, 2−6, and 3−4 in child 2), a pair of new valid structures is yielded.

Each of the two offspring structures inherits certain fragments from both of its parent. The offspring's validity is guaranteed by strictly complying with the valence rule. Because the partition of skeletal atoms, the assembly of fragments, and the saturation of intermediate structures are done in a random way, full coverage of a given constitution space is guaranteed and "blind spots" are avoided. It is unnecessary to verify the connectivity of an offspring structure, as the structure evaluation process will automatically crowd out those disconnected structures due to their poor fitness scores.

The crossover operator can be refined by a number of parameters: the *Match mode* defines how to select parent structures from the population pool; the *Partition scale* specifies at what extent the parent is allowed to spoil; and the *Partition mode* decides in which way to cleave the parent when doing the crossover operation. It has been noticed that different partition modes lead to different search inclinations.

It became necessary to introduce these parameters due to our observation that if this three-step process of cleaving, exchanging, and merging is performed in a totally random way, a large number of unconnected small segments may arise in the intermediate steps, and some obligatory substructures may be broken.

The initial purpose of a partition strategy is to make sure that at least one fragment cluster is connected (and as a result the bonds in this cluster preserved) so that only a few fragments are generated overall and the process of reconnecting them becomes computationally efficient. The procedure is to first select an arbitrary atom in parent 1 as the seed atom and then to start from there a breadth-first or depth-first walk in the structure until the number of visited atoms reaches the predecided limit. Atoms covered by the traverse path form one cluster. The rest of the atoms compose another cluster.

Breadth-first-search-based partitioning tends to restrict the structure modification in a small range, making the crossover function to act locally. On the contrary, depth-first-search-based partitioning enables the crossover to have global influence.

Niche Searcher. Regular EA is known to be strong in global search while weak in local search.[8,25] To remedy this shortcoming, an EA may be combined with an efficient local search technique. Our EA implementation is integrated with a genetic operator for vicinity search. For a specified root structure, all structures that could be generated from the root by a single mutation run construct the proximate neighborhood of the root structure. Vicinity search exhaustively checks every structure in such a niche and returns the best one. Vicinity search can be thought of as a self-adjusting process of the selected structure during its lifetime.

Within an EA run, a niche searcher is applied to a few outstanding candidates. These elite structures are intensively refined by learning from their neighbors. As a result, a gather of avant-garde is formed which is far ahead of the rest of the population. Via recombination and selection, the population tends to share the achievement of the model structures

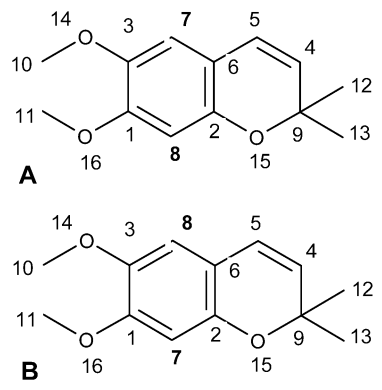


Figure 5. Two candidate structures for the molecule α-preconene with the correct one as A. The two similar structures differ only in the labeling and consequently the carbon shifts of atoms 7 and 8, as the labeling follows the descending sequence of the chemical shifts of carbon atoms in the molecule (see Table 1). Structure A has a fitness score of 3000, and structure B has a score of 2900. The structure relabeler is designed to reshuffle the chemical shifts for a group of equivalent carbon atoms in the molecule, preventing the search from being trapped in a local optimum.

Table 1. Carbon Chemical Shifts for a Sample Compound with Molecular Formula $C_{13}H_{16}O_3$

carbon no.	HCount	carbon shift (ppm)	carbon no.	HCount	carbon shift (ppm)
1	0	150.0	8	1	101.4
2	0	147.6	9	0	76.4
3	0	143.6	10	3	56.9
4	1	128.6	11	3	56.3
5	1	122.6	12	3	28.1
6	0	113.4	13	3	28.1
7	1	110.0			

very rapidly. Combining the refined local sampling into EA cycle—sending elite structures ahead by niche search and then catching them up by genetic reproduction—has been observed speeding up the progress of global optimization for the problem under study.

Some supplementary features of our EA implementation include the inverse selector, the population filter, and the structure relabeler. When an inverse selector is used, the better a candidate is, the less chance it has to take part in reproduction. A structure relabeler is a tuning operator changing the labeling of the carbon atoms of an equivalent class. Figure 5 gives two candidate structures for α-preconene ($C_{13}H_{16}O_3$). Structure A, the correct one, has a fitness score of 3000, while the second structure has a score of 2900 (please refer to the score system section below). A structure relabeler operates on a candidate (e.g. structure B here) in an attempt to find a fitter structure. The structure relabeler is supposed to be activated at late evolutionary stage and work on the best structures found so far. The population filter makes sure that copies of any structure in a population are within a threshold, called the *inhibitive value*. These supplementary operators are designed to monitor the evolution status and guide the search direction.

3.3. Fitness Function (Scoring System). The fitness function is the primary place in which an EA is tailored to a specific problem. The fitness function used for the evaluation of candidate structures is built of a suite of judges each representing a contribution of a certain optimization criterion. The EA implementation shares the same fitness

Table 2. Construction of the Fitness Function for Molecule α-Preconene

judge	constraint entries	entry no.	max score
HMBC	1-7, 1-8, 1-11, 2-5, 2-7, 2-8, 3-7, 3-8, 3-10, 4-12, 5-8, 6-4, 6-5, 6-7, 9-5	15	1500
HHCOSY	4-5, 5-4	2	200
HOSE code	one entry for every carbon atom	13	1300
Target Score: 3000			

function with other stochastic algorithm modules in the CASE system SENECA. Readers are referred to ref 3 for a detailed description of the judge design aspect. For NMR-based structure elucidation, the spectra judges handling data from 1D ^{13}C NMR, HHCOSY, HMBC, and HSQC experiments are routinely used.

Some other general-purpose constraints based on knowledge of chemical structure validity are also provided to enhance the resolving power of the fitness function. One example is the Bredt's rule judge, ensuring that no bridgehead atom in a multibridges ring system is involved in a double bond. Since judges all share the same API, it is easy to write and add a judge to the system to include a new kind of constraint.

The construction of the fitness function is quite straightforward. Taking α-preconene as an example for an unknown compound, three types of constraints, HHCOSY, HMBC, and HOSE code are provided, and three kinds of judges are employed, *HHCOSYJudge*, *HMBCJudge*, and *HOSECodeJudge*. The ^{13}C NMR spectrum is also available, but no explicit 1D spectrum judge is needed because the information inferred from DEPT 90 and DEPT 135 experiments is digested directly by the structure representation and preserved during the evolutionary process. *HHCOSYJudge* holds 2, *HMBCJudge* 15, and *HOSECodeJudge* 13 constraint entries. Satisfaction for a particular constraint entry leads to granting a predefined number of points for the candidate. This number can be individually configured, but is generally set to a value of 100. Due to this scheme, the maximum achievable fitness score for a candidate structure can easily be calculated by summing up the points earned by this structure from each constraint entry in all the judges used. Table 2 illustrates the configuration of the fitness function for the molecule α-preconene.

A fine-tuning taking into account probabilities, by which a certain spectral feature is observed, can be applied to the scoring of a judge. The HMBC judge, for example, grants 100 points, if a constraint entry can be explained by two- or three-bond CH correlations, whereas the rarer four- or five-bond CH correlations yield only 5 points. This configuration makes the search biased to the direction of the user's preference. When the maximum possible fitness score is not achieved by the best-so-far structures after sufficient generations while all judges except the HMBC judge are met, the algorithm acknowledges the existence of four- or five-bond CH correlations.

In an EA design, it is necessary for the fitness function to be able to distinguish structures even when they are extremely similar. In our EA implementation, the fitness function can also be configured to be the product instead of the sum of the scores from different judges. The product-formed fitness function is more discriminating than a sum-formed one as the former has more scoring combinations. This EA implementation still has room to improve the fitness function construction. For example, more sensitive scoring criteria should be adopted—like a *HOSECodeJudge* based on improved multisphere HOSE codes—and new, more discriminating judge types should be taken into account.

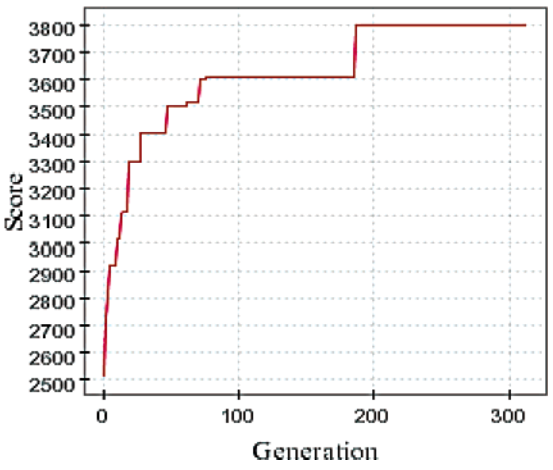


Figure 6. Evolving curves of an unknown ($C_{15}H_{28}O_2$) without parameter tuning. Calculations was performed on a computer running Windows XP equipped with a Pentium 3, 500 MHz CPU and 256 MB RAM. The fitness function involved HHCOSY, HMBC, and HOSE code judges.

It is possible to distinguish relative contributions of various judges to the best-so-far structures, for example to the upper one-fifth of the population. A vector of weighting coefficients for all judges can be defined. At first, all judges have identical coefficients. For every 20 generations, for example, the contributions of different judges are calculated. If constraints from one judge are satisfied by a higher percentage, it will be assigned with a decreased coefficient, and another poorly satisfied judge will increase its coefficient to maintain the sum of all coefficients as 1. This mechanism has been proven helpful for the algorithm to balance the search trend and avoid local convergence. Another way to make use of the weighting coefficients is to stipulate different weighting schemes for different parallel EA runs, so that each EA run follows a different route to achieve the final target. The rate of success in finding all hit structures is thus improved.

3.4. Diversity-Driven Parameter Control. The major challenge for an EA approach is premature convergence. Figure 6 is a typical evolution curve of an unknown ($C_{15}H_{28}O_2$) without parameter tuning. It is shown that after around 80 generations, the average fitness of the population had become very close to that of the best candidate and the evolution almost entirely stagnated. The impact of the number of generations on the likelihood of losing population diversity was analyzed. It is expected that the decreasing size of the search step should be corresponding to the increasing resemblance of the top candidates to the optimum. If the reproduction operators decrease their step sizes faster than the rate by which the candidates approach the optimum, the search process may stagnate, since the step sizes become too small to make the candidates sufficiently different.

Taking into account that different stages in an evolutionary search require different step sizes, and different fitness

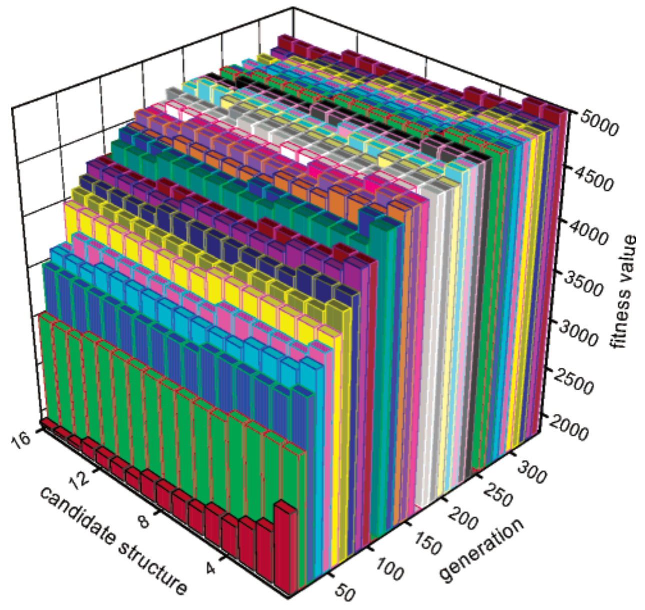


Figure 7. Evolution of monochaetin ($C_{18}H_{20}O_5$). A population of 16 structures evolves over about 350 generations. In the end of the evolutionary process, about 10% of the candidate structures have reached the maximum achievable score of 5000 points.

function component require different step sizes, a dynamic step size control strategy is adopted in this EA implementation, which applies the information accumulated so far in evolutionary search. A large step size encourages long-range search and makes escaping from a poor local optimum easier, while a smaller step size is apt to exploit a small region. For an unknown structure to be optimized, basically it is hard to predict when to sample a large space and when to explore a small region. We propose that the measure of population diversity and the distance of best-so-far structure to the target structure could be used to guide the tradeoff between coarse-grained sampling and fine-grained exploration.

The mechanism for diversity-guided step size control is to define a metric over the constitution space and ensure that the distance between any parent−offspring pair involved in a genetic operation is greater than a specified minimum threshold. This minimum threshold is subjected to change based on its judgment on the current search status (e.g., how close the search is to its target) and population distribution. Several computationally affordable similarity index and fingerprint methods are used in this implementation as the metric measures. At the beginning of the search, the constitutional space is sampled over a relatively coarse grid, with the mesh size equal to the metric. As the search progresses, the grid size is gradually reduced such that adjacent structures are also considered. This mechanism is a compromise between speed and efficiency, because we can only make sure that an offspring is sufficiently different from its parent. It is very expensive, if not impossible, to maintain a distance threshold between all candidate structures in a population because of the intractability to obtain an analytical representation of the constitutional space.

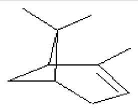
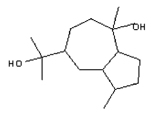
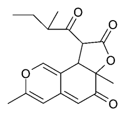
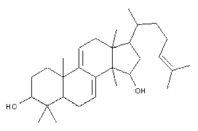
The control of parameters is performed in different levels, from a single individual, a pool of individuals (population), to a set of populations manipulated by the algorithm, which will be discussed in a separate paper.

Table 3. Empirical Parameter Settings for Sample Problems

samples	$C_{10}H_{16}$	$C_{15}H_{28}O_2$	$C_{18}H_{20}O_5$	$C_{30}H_{48}O_2$
population size	8	36/48	36/48	120/160
replacement rate	100	100	100	100
killing rate	0	0	0.1	0.2
mutation rate	0.5	0.25	adaptive	0.30
niche search rate		0.25	1[a]	0.30
crossover rate	0.5	0.50	adaptive	0.40
mutation strength	1	1	1	1
match mode	random	random	difference	random
partition mode	random	depth-first (df)	breadth-first (bf)	bf/df = 7/3
partition scale		4-6	4-8	4-11

[a] Niche search works on the top two structures and happens when the population scale is to change (from 36 to 48) after every five generations of no improvement.

Table 4. Performance Overview of the EA Implementation with Parameter Settings Given in Table 3[a]

structure	molecular formula	points visited vs general	generations	solutions	calculation time (s)
	$C_{10}H_{16}$	48/4305	6	1	<1
	$C_{15}H_{28}O_2$	2088/?	60	1	40
	$C_{18}H_{20}O_5$	2600/?	70	1	120
	$C_{30}H_{48}O_2$	6400/?	48	6[b]	120

[a] Calculations were performed on a computer running Windows XP equipped with a Pentium 3, 500 MHz CPU and 256 MB RAM. Results are taken as an average of 20 runs. The fitness function involved HHCOSY, HMBC, and HOSE code judges. Calculation times are given as TRS50, defined as the mean time in which the 50% most successful processes reach the optimum. As for any other stochastic method, calculations have to be run repeatedly in order to provide a statistical backing of the result and to increase the probability of finding all correct solutions. These calculations can be carried out in a parallel manner on a network of commodity-type computers, as has been shown for our simulated annealing implementation in an earlier paper.[3] [b] In all, six structures are found after 20 runs, which are consistent with those found by a deterministic approach.[27]

4. RESULTS

First we take monochaetin[26] ($C_{18}H_{20}O_5$) as the proof-of-concept example. The spectra information comes from ^{13}C NMR, HHCOSY, and HMBC. The HHCOSY judge has 8 constraint entries and HMBC 24. Together with the contribution of the HOSE code judge (18 entries), the maximum achievable fitness score is 5000. The three judges have identical and constant weighting coefficients in all EA runs. No other judges are employed to make the problem simple.

To reflect the process of structure evolution, a small population size (16 individuals) is selected, and only two genetic operators are used, crossover and mutation. The probabilities of crossover and mutation are controlled

$$P_{\text{crossover}} = 0.5\sqrt{1 - S_{\text{best}}/S_{\text{goal}}} \quad (1)$$

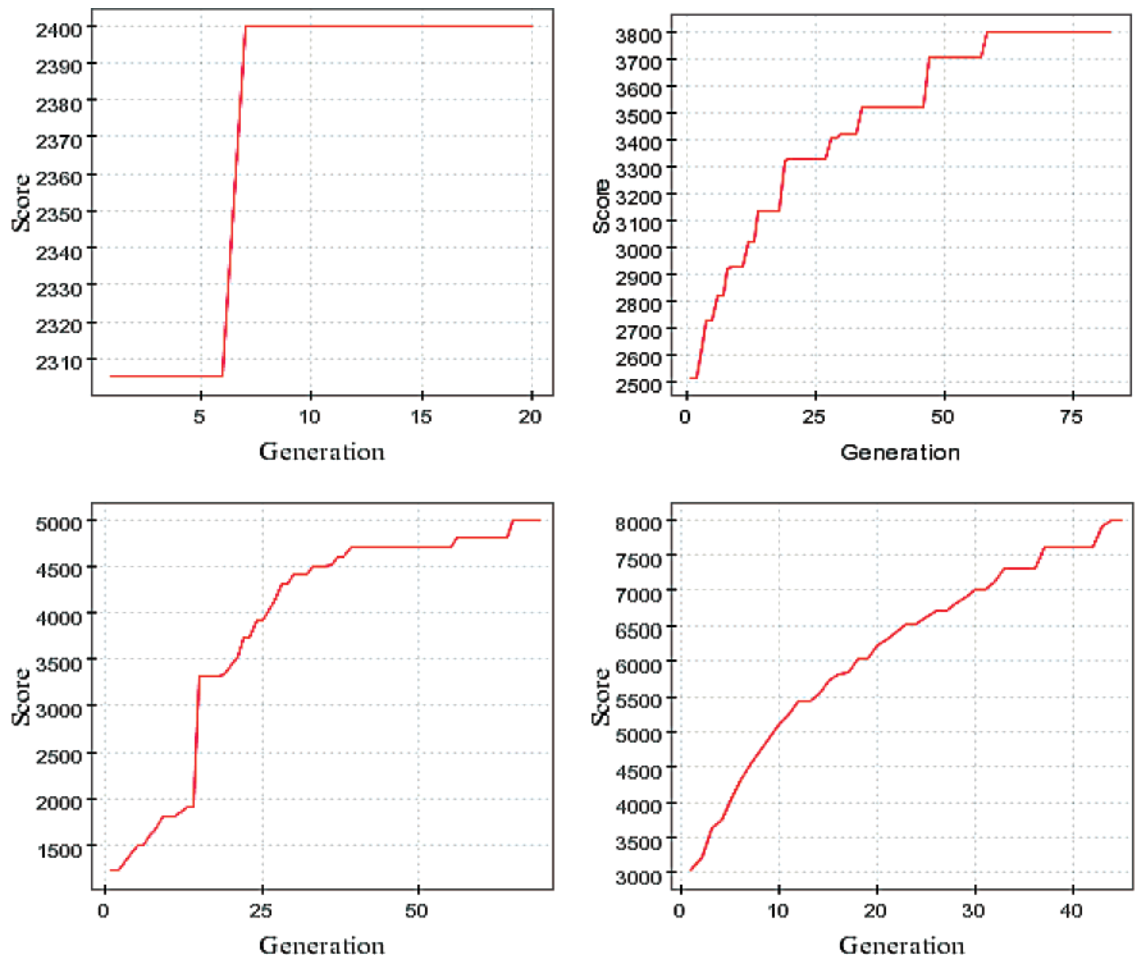


Figure 8. Evolving curves of the four examples given in Table 3. Calculations were performed on a computer running Windows XP equipped with a Pentium 3, 500 MHz CPU and 256 MB RAM. The fitness function involved HHCOSY, HMBC, and HOSE code judges.

where S_{best} stands for the fitness score of the best-so-far

$$P_{mutation} = 1 - P_{crossover} \quad (2)$$

structure and S_{goal} is the fitness score of the target structure(s) for the unknown. For crossover operation, the value of *Partition scale* ranges from 4 to 8 (one-third of the skeleton atom number 23) and is determined according to

$$V_{partition} = (V_{max} - V_{min})(1 + \mathrm{Dev}/\mathrm{Dev}_{empirical}) \quad (3)$$

Here, V_{max} and V_{min} are maximum and minimum allowed values of the partition scale; Dev is the standard deviation of the fitness scores of the last population; and $\mathrm{Dev}_{empirical}$ is the empirical maximum standard deviation of the fitness scores of a population (in this case 200). $\mathrm{Dev}_{empirical}$ is obtained by calculating the average of the standard deviations of a set of initial populations.

In each generation, a temporary population of 16 new structures is created through reproduction operations. Among the current and temporary population (32 structures in all), 16 are selected to form the offspring population with the top two structures always being preserved. Selection is through either the *Tournament Selector* or the *Ranking Order Selector*.

The result is illustrated in Figure 7. One hit is found after about 350 generations within a 3 min run. There are about 5600 points sampled in the constitution space, which is a tiny amount compared to the huge number of constitutional isomers for $C_{18}H_{20}O_5$. Notably, this result is obtained with a less appropriate population scale, and no other constraints are introduced or other parameters optimized.

Several structure elucidation examples of increasing molecular size are shown in Table 3, Table 4, and Figure 8 to evaluate the algorithm's performance. Table 3 gives typical parameter settings for 4 compounds of different size. The task of Monochaetin structure elucidation is approached again using a mechanism of population scale control. An EA run adjusts its population following the procedures below.

1. Start the EA with population size as 36 and iterate steps 2−6.

2. If after 5 generations no better solutions appear, delete one-third of the worst candidates and expand the population size to 48. New individuals are created by doing a niche search around the top 5 structures in the population.

3. EA now runs with the population size as 48.

4. If after 5 generations no new solutions are found, set the population size back to 36, the removed individuals being half from good ones and half from bad ones (the best solutions are always preserved).

5. EA now runs with the population size as 36.

6. For every 20 generations, an isomorphism check is done to reshuffle the population.

Averagely, with this scheme, 50% of EA runs can converge to the optimum after sampling 3000 points in constitution space; other runs take longer time, but almost all are able to find the solution.

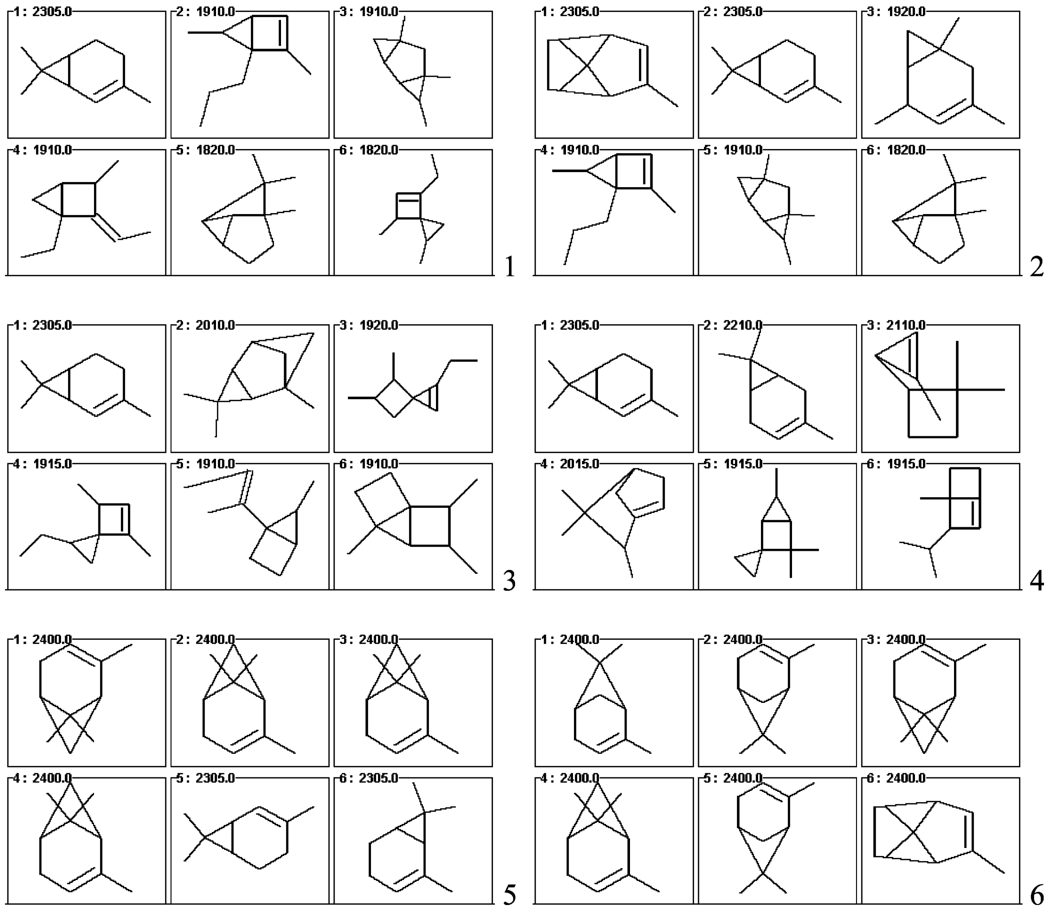


Figure 9. Evolution of α-pinene ($C_{10}H_{16}$). The top six structures in six successive generations are shown (constraints: molecular formula, DEPT 135/90, HMBC, and carbon chemical shifts).

In many cases, instead of one optimum structure, a number of structures exist satisfying all given inputs. In such a case, finding all solutions cannot be achieved with a relatively small population scale. Therefore, a set of EA runs needs to be carried out before any conclusions are drawn. For example, a single EA run configured as in Table 3 cannot find all six structures for compound 4 (polycarpol), but running the algorithm 20 times will succeed. This is a well-known property of stochastic optimization schemes, which is why frequently a statistic backing of the result is created by collecting the outcome of multiple optimization runs performed either in a parallel or a sequential manner.

Compared to an earlier deterministic structure elucidation module created in our group,[27] both stochastic algorithms, simulated annealing[3] as well as the evolutionary algorithm described in this paper, scale much smoother in their computation times versus the number of heavy atoms in the problem set.

Besides, we are currently testing a newly designed multipopulation-based evolution scheme to maintain EA's coverage of the problem space, especially for unknowns with huge search space. The new scheme inherits a concept of a two-level evolution: a set of peripheral EA threads run independently with different configurations. For an interval of 60 s, each EA thread contributes a number of (cloned) good structures to generate the population of the core EA thread. The peripheral EA runs find local optima in different regions, and the core EA thread aims to refine the good structures by niche searches. This scheme is currently being tested with several larger compounds.

5. SUMMARY

An evolutionary algorithm using graph-based data structure has been presented, having been applied to structure elucidation of organic and bioorganic compounds. The algorithm provides a suite of robust graph operators to propel the evolution of molecular structures toward a set of desired properties. The graph data structure facilitates efficient genetic manipulation and exempts from the transformation between genotype and phenotype of the candidate solution. A strong control of their parameters enables the genetic operators to adjust their behavior and achieve higher search efficiency. To the best of our knowledge, we have considerably increased the size of molecular constitutions treatable with evolutionary algorithms.

The implemented search engine now is part of the CASE program SENECA, and its performance is demonstrated by solving real-world structure elucidation problems.

ACKNOWLEDGMENT

The SENECA project was supported by the German Research Council.

APPENDIX

The evolution of α-pinene ($C_{10}H_{16}$) is shown in Figure 9.

REFERENCES AND NOTES

(1) Munk, M. E. Computer-Based Structure Determination: Then and Now. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 997−1009.

(2) Faulon, J. L. Stochastic Generator of Chemical Structure. 2. Using Simulated Annealing To Search the Space of Constitutional Isomers. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 731−740.
(3) Steinbeck, C. SENECA: A Platform-Independent, Distributed, and Parallel System for Computer-Assisted Structure Elucidation in Organic Chemistry. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1500−1507.
(4) Meiler, J.; Will, M. Automated Structure Elucidation of Organic Molecules from C-13 NMR Spectra Using Genetic Algorithms and Neural Networks. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1535−1546.
(5) Jaspars, M. Computer Assisted Structure Elucidation of Natural Products Using Two-Dimensional NMR Spectroscopy. *Nat. Prod. Rep.* **1999**, *16*, 241−247.
(6) Williams, A. Recent Advances in NMR Prediction and Automated Structure Elucidation Software. *Curr. Opin. Drug Discovery Dev.* **2000**, *3*, 298−305.
(7) Elyashberg, M. E.; Blinov, K. A.; Williams, A. J.; Martirosian, E. R.; Molodtsov, S. G. Application of a New Expert System for the Structure Elucidation of Natural Products from Their 1D and 2D NMR Data. *J. Nat. Prod.* **2002**, *65*, 693−703.
(8) Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Reading, MA, 1989.
(9) Sheridan, R. P., Kearsley, S. K. Using a Genetic Algorithm To Suggest Combinatorial Libraries. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 310−320.
(10) Agrafiotis, D. K. Stochastic Algorithms for Maximizing Molecular Diversity. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 841−851.
(11) Brown, R. D.; Clark, D. E. Genetic Diversity: Applications of Evolutionary Algorithms to Combinatorial Library Design. *Expert Opin. Ther. Pat.* **1998**, *8*, 1447−1459.
(12) Nair, N.; Goodman, J. M. Genetic Algorithms in Conformational Analysis. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 317−320.
(13) Frey, C. An Evolutionary Algorithm with Local Search and Classification for Conformational Searching. *Match-Commun. Math. Comput. Chem.* **1998**, 137−159.
(14) Douguet, D.; Thoreau, E.; Grassy, G. A Genetic Algorithm for the Automated Generation of Small Organic Molecules: Drug Design Using an Evolutionary Algorithm. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 449−466.
(15) Clark, D. E. An Overview of Evolutionary Algorithm Applications in Computer-Aided Molecular Design. *Abstr. Pap.*—*Am. Chem. Soc.* **2001**, *221*, 25-COMP.
(16) Globus, A.; Lawton, J.; Wipke, T. Automatic Molecular Design Using Evolutionary Techniques. *Nanotechnology* **1999**, *10*, 290−299.
(17) Nachbar, R. B. Molecular Evolution: Automated Manipulation of Hierarchical Chemical Topology and Its Application to Average Molecular Structures. *Genet. Program. Evol. Mach.* **2000**, *1* (1/2), 57−94.
(18) Meiler, J.; Will, M. Genius: A Genetic Algorithm for Automated Structure Elucidation from C-13 NMR Spectra. *J. Am. Chem. Soc.* **2002**, *124*, 1868−1870.
(19) Bremser, W. HOSE—A Novel Substructure Code. *Anal. Chim. Acta* **1978**, *103*, 355−365.
(20) Bremser, W. Expectation Ranges of 13-C NMR Chemical Shifts. *Magn. Reson. Chem.* **1985**, *23*, 271−275.
(21) Steinbeck, C.; Kuhn, S.; Krause, S. NMRShiftDB—Constructing a Chemical Information System with Open Source Components. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1733−1739 (http://www.nmrshiftdb.org, visited on June 2003).
(22) Weininger, D. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31−36.
(23) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 97−101.
(24) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493−500.
(25) Mitchell, M. *An Introduction to Genetic Algorithms*; MIT Press: Cambridge, MA, 1996.
(26) Christie, B. D. The Role of Two-Dimensional Nuclear Magnetic Resonance Spectroscopy in Computer-Enhanced Structure Elucidation. *J. Am. Chem. Soc.* **1991**, *113*, 3750−3757.
(27) Steinbeck, C. Lucy—A Program for Structure Elucidation from NMR Correlation Experiments. *Angew. Chem., Int. Ed. Engl.* **1996**, *35*, 1984−1986.
(28) Sundaram, A.; Venkatasubramanian, V. Parametric Sensitivity and Search-Space Characterization Studies of Genetic Algorithms for Computer-Aided Polymer Design. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 1177−1191.

CI034132Y