Using a Genetic Algorithm To Suggest Combinatorial Libraries

Robert P. Sheridan* and Simon K. Kearsley
RY50S-100, Merck Research Laboratories, P.O. 2000, Rahway, New Jersey 07065

Received November 21, 1994®

In combinatorial synthesis, molecules are assembled by linking chemically similar fragments. Since the number of available chemical fragments often greatly exceeds the number of distinct fragments that can be used in one synthetic experiment, choosing a subset of fragments becomes problematical. For example, only a few dozen distinct primary and secondary amines have ever been reported to have been used in constructing a library of peptoids (oligomers of N-substituted glycine), while there are several thousand suitable primary and secondary amines that are commercially available. If a combinatorial library is to be constructed with a particular biological activity in mind, computer-based structure−activity methods can be used to rationally select a subset of fragments. In principle one would computationally generate every possible molecule as a combination of fragments, score each molecule by the likelihood of its being active, and select those fragments that occur in high-scoring molecules. For many cases there are too many combinations to take this exhaustive approach, but genetic algorithms can be used to quickly find high-scoring molecules by sampling a small subset of the total combinatorial space. In this paper we demonstrate how a genetic algorithm is used to select a subset of amines for the construction of a tripeptoid library. We show three examples. In the first example, the scoring is based on the similarity of the tripeptoids to a specific tripeptoid target. Since the target itself can be generated in this example, we have an opportunity to experiment with the protocol of our genetic algorithm. In the second example, scoring is based on the similarity to two tetrapeptide CCK antagonists. In the third, scoring is done by a trend vector derived from activity data on ACE inhibitors. In all cases we show that the genetic algorithm can find, in a modest amount of computer time, high-scoring peptoids that resemble the targets.

INTRODUCTION

The idea behind combinatorial synthesis (reviewed in ref 1 and 2) is that large libraries of chemical compounds can be created by joining a *basis set* of fragments into short oligomers. These libraries are then to be screened against a variety of biological assays. One of the problems in combinatorial synthesis is that the number of chemical fragments that *could* be used is very much larger than the number of fragments that actually can be used as a basis set in a given experiment. For instance, peptoids,[3−5] polymers of N-substituted glycine, are constructed by joining amines with linker fragments. No basis set of more than a few dozen or so distinct amines has ever been reported in the construction of a peptoid library.[3,5] However, the number of chemically suitable primary or secondary amines in the Fine Chemicals Directory (FCD), a database of commercially available compounds,[6] is at least a few thousand. Thus, the chemical space of possible peptoids can be sampled only sparsely by a single basis set. In practice, fragments for a basis set are chosen arbitrarily or because they contain chemical groups associated with activity on certain classes of receptor (for instance see ref 5). It would be extremely useful to have a method to rationally select a small number of fragments from a large set. If we decide that we want to construct a library aimed at a particular biological activity, computer-based SAR methods provide a way to do the selection. We would be covering only a small part of chemical space as before, but hopefully it will be the part of space most likely to contain active molecules.

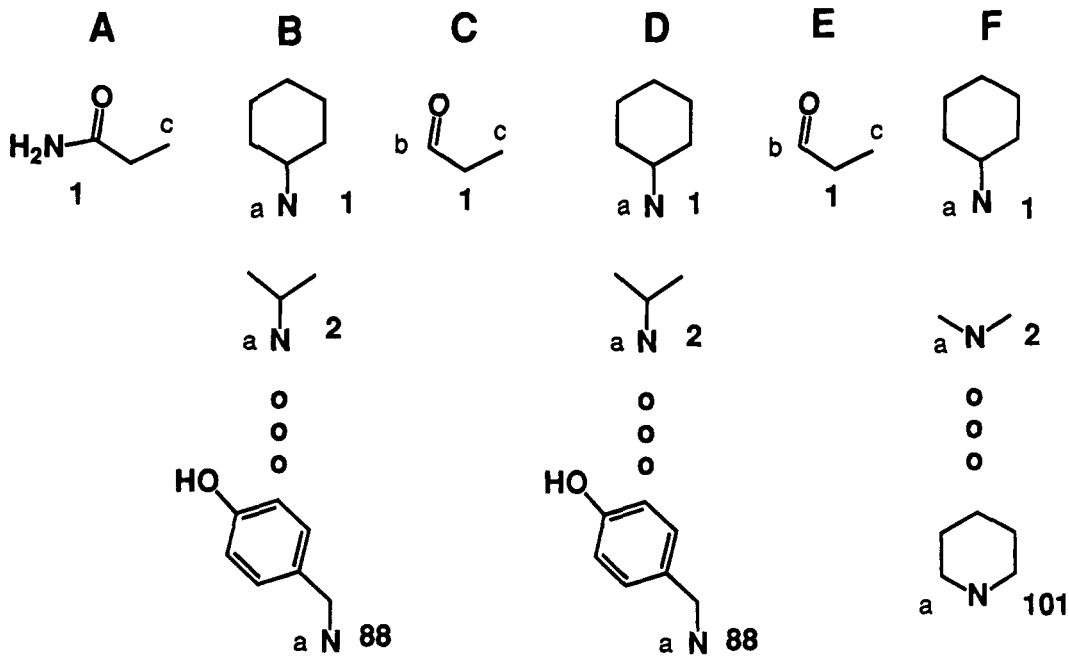
Similarity probes[7] and trend vectors[7,8] are useful topological SAR methods that can assign to a molecule a likelihood of its having a particular biological activity. Both have proven effective in a number of projects for selecting active molecules from large databases.[7−9] Typically the "hit rate" is 5- to 40-fold higher than from random screening. Often the active molecules are novel, demonstrating that the methods can generalize across chemical classes. Consider the following thought experiment: We decide to make a tripeptoid library where the first and second position are occupied by a primary amine and the third by either a primary or secondary amine. On the computer, construct the connection table for every possible peptoid, calculate its descriptors, and score it against a similarity probe or trend vector. Save the peptoids with the highest scores. See what amines are in the high-scoring molecules. These are the amines most likely to confer activity, the ones we want to use in our basis set. This exhaustive approach is impractical because, with a few thousand amines possible at each position, there are tens of billions of combinations. Fortunately, genetic algorithms are very efficient computational methods for searching large combinatorial spaces.

Genetic algorithms are the computational analog of Darwinian evolution. (Useful general refs are 10 and 11.) Individual entities are represented as a "genome", a linear series of genes. Each gene may have one of a set of values, each value being an "allele". In some genetic algorithms, one starts with a population of a given size, with each individual in the population having a randomly selected allele in each gene. The individuals are then scored by some "fitness function". A new population is constructed from the old through three mechanisms. First, some individuals can be copied intact from the old population. Second, a new individual can be created by "mutating" one or more genes of an individual in the old population. Third, a new

® Abstract published in *Advance ACS Abstracts*, March 1, 1995.

0095-2338/95/1635-0310$09.00/0

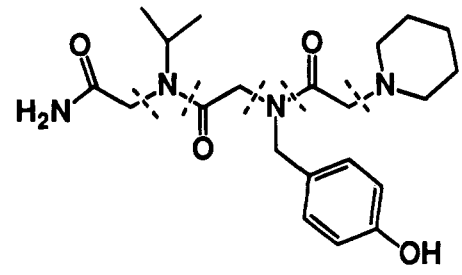
Fragment libraries:



Fusion instructions to produce a tripeptoid:

First fragment	A	1		
Fusion 1	B	2	c-a	blank-keep
Fusion 2	C	1	a-b	blank-blank
Fusion 3	D	88	c-a	blank-keep
Fusion 4	E	1	a-b	blank-blank
Fusion 5	F	101	c-a	blank-blank

Example:



Genome: 1 2 1 88 1 101

Figure 1. An example of how a tripeptoid is build from six residues. Each residue has a fragment library associated with it. In each fragment, selected atoms are marked by an fusion type (a, b, c). The fusion instructions describe the sequence for joining the fragments. In this example, the first instruction is to take fragment 1 from library A. The second instruction is to take a fragment 2 from B and covalently bond an atom in the A fragment with fusion type c to an atom in the B fragment with a fusion type a. The following "blank" indicates that the fusion type of the bonded atom in the A fragment is to be erased so that it is not available for further bonding. The "keep" indicates that the amine nitrogen in the B fragment is to keep its type. We now have the A−B dimer. The third instruction is to pick a fragment 1 from library C and bond an atom in A−B with fusion type a to an atom in the C fragment with fusion type b. Both atoms joined by the new bond are blanked. The stepwise fusion continues in this way until all the fragments are joined. The final result is shown at the bottom of the Figure.

individual can be created by "crossing-over" the genomes of two "parents" from the old population. Higher scoring individuals have a higher probability of passing their genes to the new population by any of these mechanisms. The new population is scored in turn and the cycle continues. Typically the average and maximum score of the population rise with each generation until they converge to some maximum value.

Genetic algorithms are stochastic. That is, random selections are involved at nearly all stages and the results depend on particular random number sequences. There is no guarantee that the global maximum will be found, or, if there is more than one good solution, that all will be found. However, for most large combinatorial problems, better solutions will be found faster with a properly implemented genetic algorithm than with a systematic search or with a

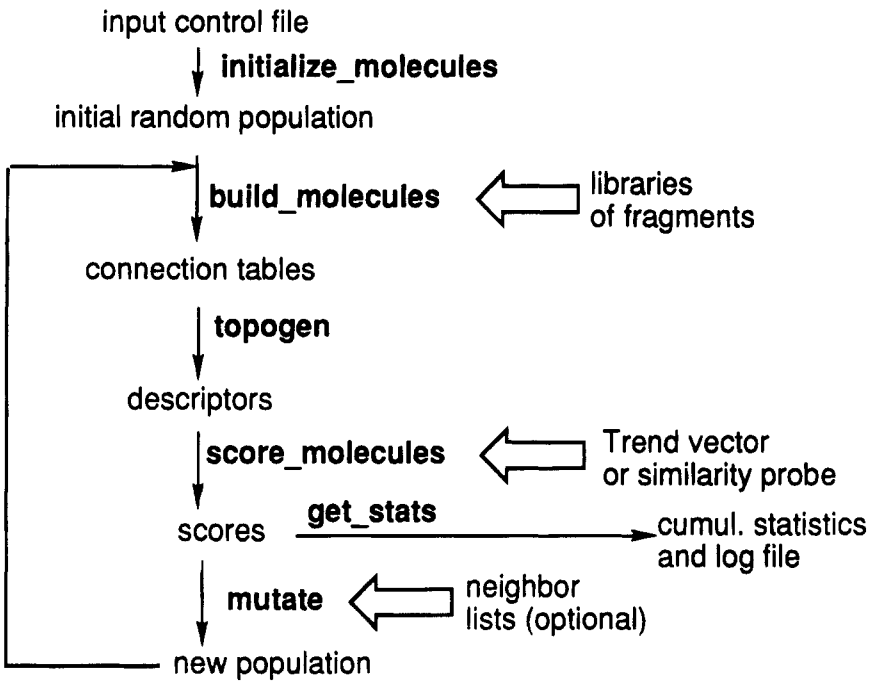


Figure 2. A schematic diagram of the genetic algorithm script. FORTRAN or C programs are in bold.

random search without selection. (See Chapter 1, ref 10.)

This paper describes how we have applied genetic algorithms to find the highest scoring peptoids given a large universe of amines and demonstrates that the technique is very effective in finding high-scoring peptoids out of a large number of combinations.

METHODS

Assembly of Molecular Fragments. Imagine that we have a set of fragments that are to be joined into a molecule. Within each fragment, the atoms that may form a covalent bond are marked by some arbitrary letter, which we will call a *fusion type*. A set of fragments with similar fusion types may be collected in a *fragment library*. There will be one library associated with each *residue* of the whole molecule. The libraries for any two residues may or may not be the same. A set of *fusion instructions* tells what sequence to follow to create the connection table of the whole molecule. An example of the libraries and instructions for forming a *tripeptoid* are shown in Figure 1. In the nomenclature of genetic algorithms, the genome of a tripeptoid consists of six genes A−F, three of which have more than one allele.

Scoring Molecules. For the scoring methods presented here, the connection table of each molecule is parsed into a set of topological descriptors. Descriptors we have found useful are the atom pair[7] and the topological torsion.[12] These descriptors have two important properties. First, they are easily calculated from connection tables. Second, they are general enough to apply to diverse chemical classes but specific enough in the aggregate to distinguish among closely related isomers. In this paper, we will use the atom pair exclusively, but other types of descriptors could be used. The atom pair descriptor consists of a pair of atom types separated by a specific number of bonds. Atom type includes information about elements, number of non-hydrogen neighbors, and number of π electrons.

The first scoring method uses chemical similarity. Given a set of descriptors for two molecules, one may calculate the similarity of the molecules as the number of descriptors they have in common normalized by the average number of descriptors in the two molecules.[7] Similarity may range from 0 (nothing in common) to 1 (identity). In many applications one has a *probe* against which a number of molecules are measured. A probe need not be a single molecule but a descriptor average of two or more. The idea behind similarity scoring is that molecules that are similar to the probe are likely to have similar biological activity.

The second scoring method is trend vector prediction. Trend vector analysis[7,8] is a method of summarizing the correlation of descriptors with biological activity for large training sets of molecules, such as those produced by large-scale screening. A Monte Carlo method is used to ensure that the correlation is statistically significant. The trend vector can be used to calculate the predicted activity of a compound outside the training set. Here we use the sample-based partial-least-squares reformulation of trend vector analysis.[8] Details of how trend vectors are calculated and how predictions are made are given in ref 8. Before a prediction can be made, at least 95% of the descriptors in the compound to be predicted must be found in the training set from which the trend vector was derived. (This avoids the problem of extrapolating too far beyond the training set.) For the purposes of our genetic algorithm, wherein a score must be produced for every molecule, the score of a molecule which does not meet this criterion is zero.

Implementation of the Genetic Algorithm. We encoded the genetic algorithm as a Unix shell script that ties together several FORTRAN or C programs. The scheme is represented in Figure 2. An input control file contains parameters for the run (population size, number of generations), names of files containing the fragment libraries, etc. The program **initialize_molecules** generates a population of molecules by randomly assigning for each residue a number from 1 to N, where N is the number of fragments in the library for that residue. For example, part of the population of tripeptoids could look like:

A	B	C	D	E	F
1	1615	1	435	1	2540
1	5	1	1234	1	601
1	2134	1	456	1	301

etc.

The program **build_molecules** takes each list of genes and generates a connection table by following the fusion instructions.

The program **topogen** generates topological descriptors from the connection tables, and **score_molecules** calculates a score for each molecule, based on its descriptors, against a similarity probe or a trend vector. The program **get_stats** monitors the average, minimum, and maximum score for the population and writes the individual genomes and scores to a log file. The program **mutate** prepares the next population from the previous one.

Selection/Generation Protocols. There is no single standard method by which a genetic algorithm generates a new population from the previous one based on the scores. Methods are usually problem-specific, and we developed the program **mutate** specifically for the combinatorial synthesis problem. We incorporated two different selection/generation protocols. In the *best third* method, an elitist strategy, the highest scoring third of the population is saved. For instance, for a population of 300, the best 100 are saved; the others are summarily "killed". Three copies are made of this subset of 100. The first copy survives unchanged into the new population. For each individual in the second copy, a single residue is randomly chosen, and that residue is mutated, e.g., if residue D is chosen

$$1\ 234\ 1\ 1923\ 1\ 2560 \rightarrow 1\ 234\ 1\ 1804\ 1\ 2560$$

(Residues with only one possible allele are never chosen, since mutations there would be meaningless.) The second copy is then added to the new population. Each member of the third copy is crossed-over with another randomly chosen member of the same copy at a randomly chosen crossover point, e.g.

$$1\ 345\ 1\ |\ 1025\ 1\ 891 \rightarrow 1\ 345\ 1\ |\ 678\ 1\ 2304$$

$$1\ 1023\ 1\ |\ 678\ 1\ 2304 \rightarrow 1\ 1023\ 1\ |\ 1025\ 1\ 891$$

The third copy is then added to the new population.

In the *stochastic* method, 100 individuals from the old population of 300 are randomly selected to survive unchanged, 100 individuals are randomly selected to survive with a single mutation, and 100 individuals are generated as the crossover product of two randomly selected parents. In all cases, the relative probability of an individual i being selected depends on a linear function of its score

$$\text{prob}_i = (\text{score}_i - \text{minscore})/(\text{maxscore} - \text{minscore})$$

where minscore and maxscore are the minimum and maximum score in the old population.

We also incorporated two alternative protocols for generating mutations. In *random* mutation, any allele can mutate into any other allele with equal probability. In *neighbor* mutation, an allele can mutate only to one of the k fragments most similar to it. This option ensures that molecules mutate to closely related molecules. To use this option, one must have a list of neighbors for each allele appropriate for a given fragment library.

Another important consideration is how to ensure diversity; we do not want the population swamped with identical individuals. The user has an option of how to handle duplicates in the new population after it has been constructed: keep them, delete them, or change them into something else by single residue mutation. We consistently followed the last choice.

Fragment Libraries. Since we are using peptoids as examples, the natural unit is the amine. In the FCD there are 2507 primary amines and 805 secondary amines with the following properties:

1. Molecular weight $\leq$ 200.
2. Exactly one reactive amine per molecule. For this purpose the definition of reactive is as follows: the amine N is bonded to no more than two non-hydrogens and is bonded to no more than one unsaturated atom.
3. There are no interfering chemical groups: oximes, sulfhydryls, azides, nitros, and nitrosos.

In the FCD, there are many amines that are topologically identical but differ in salt form or stereochemistry. We did not try to eliminate topological duplicates.

There are four distinct fragment libraries for tripeptoids:

1. The set of primary amines for residues B and D (2507 entries).
2. The set of primary and secondary amines for residue F (3312 entries).
3. An "initiating" fragment corresponding to residue A (1 entry).
4. A "linker" fragment for residues C and E (1 entry).

All libraries are stored in a randomly accessible database (Miller, M. D., unpublished work) such that a fragment can be retrieved by providing its number. There are two distinct neighbor lists, one used for residues B and D and one used for residue F.

With this set of libraries there are $2507 \times 2507 \times 3312 = \sim 20$ billion possible tripeptoids.

For each fragment in a library we saved as its neighbors the $k = 10$ fragments most similar to it (using the atom pair descriptor). Because we want to avoid null mutations, topologically identical fragments do not count as neighbors.

RESULTS

We will look at three test cases wherein tripeptoids are constructed and then scored by various means.

1. Peptoid Target. The first example is used to test our genetic algorithm protocols. For this purpose, we will need to recognize the "right answer" when it appears. We chose an arbitrary tripeptoid as a target; when a tripeptoid identical to the target is generated by the genetic algorithm, its similarity to the target, i.e., its fitness score, will be 1.0. The target will be the tripeptoid where residue B is 4-fluorophenylethyamine, residue D is biphenylamine, and residue F is tyramine. There are actually two ways to produce this molecule, there being two copies of tyramine in the library for residue F.

By trial and error we found that using a population of 300 is reasonable compromise between having enough sampling over combinatorial space and not taking too much computer time. We ran the genetic algorithm with both selection

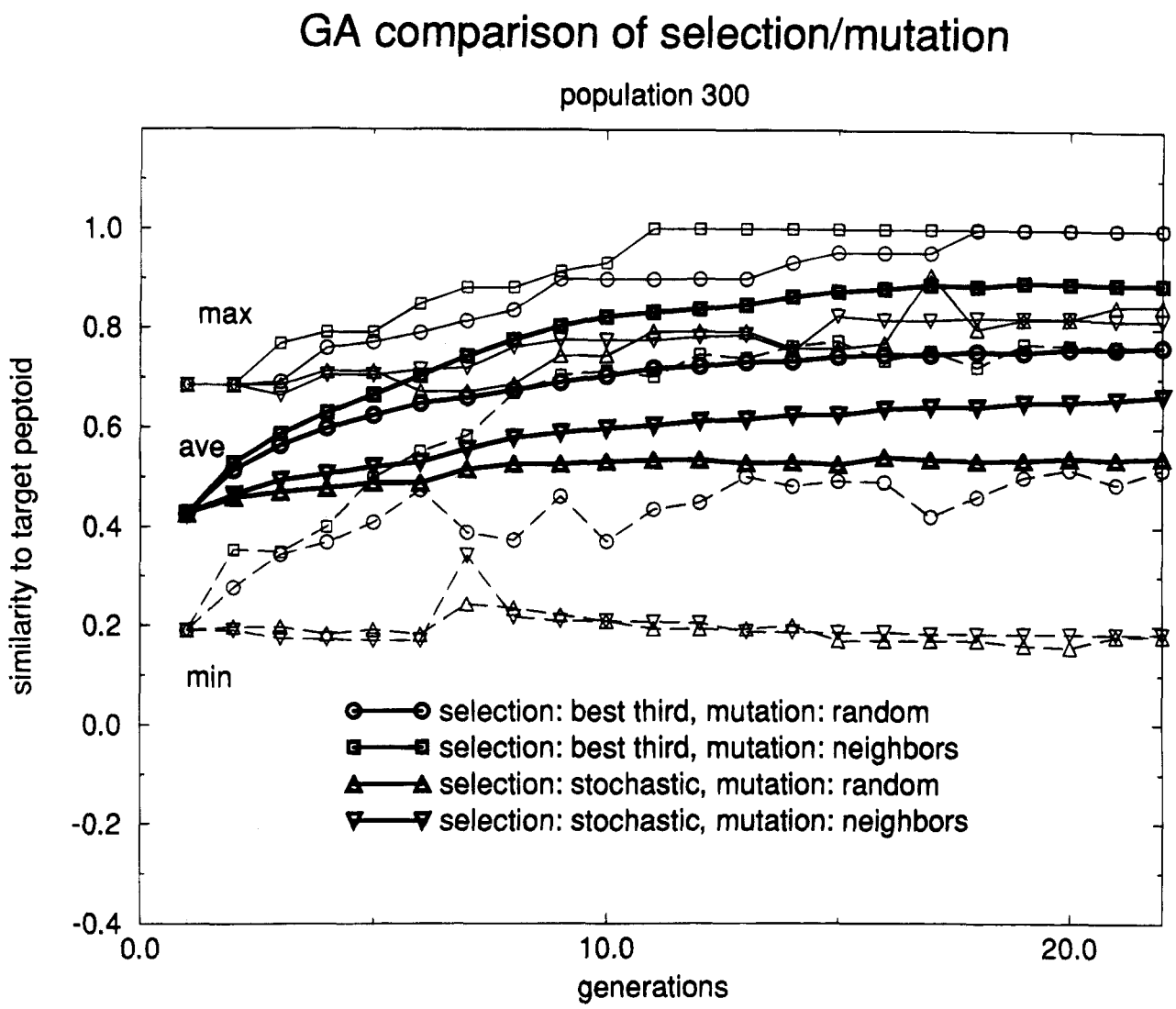


Figure 3. A graph of the minimum, average, and maximum score as a function of generation for a population of 300 against a target peptoid. Four combinations of selection method and mutation method are shown. These runs all start with the same random number seed.

methods and with both mutation methods, using the same starting random number seed. One generation for a population of 300 takes about 6 min on a Personal Iris. We monitored the maximum, average, and minimum score as a function of generation. This graph is in Figure 3. In all cases, the average score rises. Clearly, using the best third selection method is much better than the stochastic method in raising the average score more quickly for either mutation method. The average score for the former converges in <25 generations, while the latter has not converged. Also, neighbor mutation raises the average score somewhat more quickly than random mutation for either selection method.

The maximum score jumps up and down for the stochastic selection, while it monotonically increases for the best third selection, as expected for an elitist algorithm. In this run, the maximum score curve for top third selection with neighbor mutation reached 1.0, i.e., found one right answer, in 11 generations. It found both right answers at 12 generations. By generation 12 the algorithm had examined only ~3000 distinct peptoids out of 20 billion combinations. Figure 4 shows the effect of different random number seeds for this protocol. It seems the first run was lucky but not particularly so; other seeds find one answer after 13, 17, and 23 generations. The last example shows it is possible to get stuck in a suboptimal solution; it converges at nine generations to similarity = 0.90. Thus, it is useful to rerun any particular problem with different random number seeds.

It is instructive to look at the structures produced by the algorithm. Figure 5 shows the highest-scoring structure in the population as a function of generation for the run that took 23 generations to get the answer. In generation 1, the highest-scoring peptoid hardly resembles the target. At generation 5, there is still not much resemblance except for residue F. At generations 10 and 15, two out of three residues are correct. By generation 20, the molecule differs by one atom from the target. Finally by generation 23 the target is found.

Similarity to a CCK Antagonist Tetrapeptides. Our second example demonstrates the case of finding the tripeptoids that most resemble a drug target. We arbitrarily chose as a target two tetrapeptide CCK antagonists taken from the MACCS Drug Data Report (MDDR), a licensed database of about 43 000 druglike structures compiled from the open literature.[13] The structures of these are shown in the top of Figure 6. The score of each tripeptoid is taken as the similarity to the descriptor average of the two probes. As before, we used a generation size of 300, but used only the best third selection and neighbor mutation since this seemed to be the most useful protocol. We did three runs of 25 generations, each with a different random number seed.

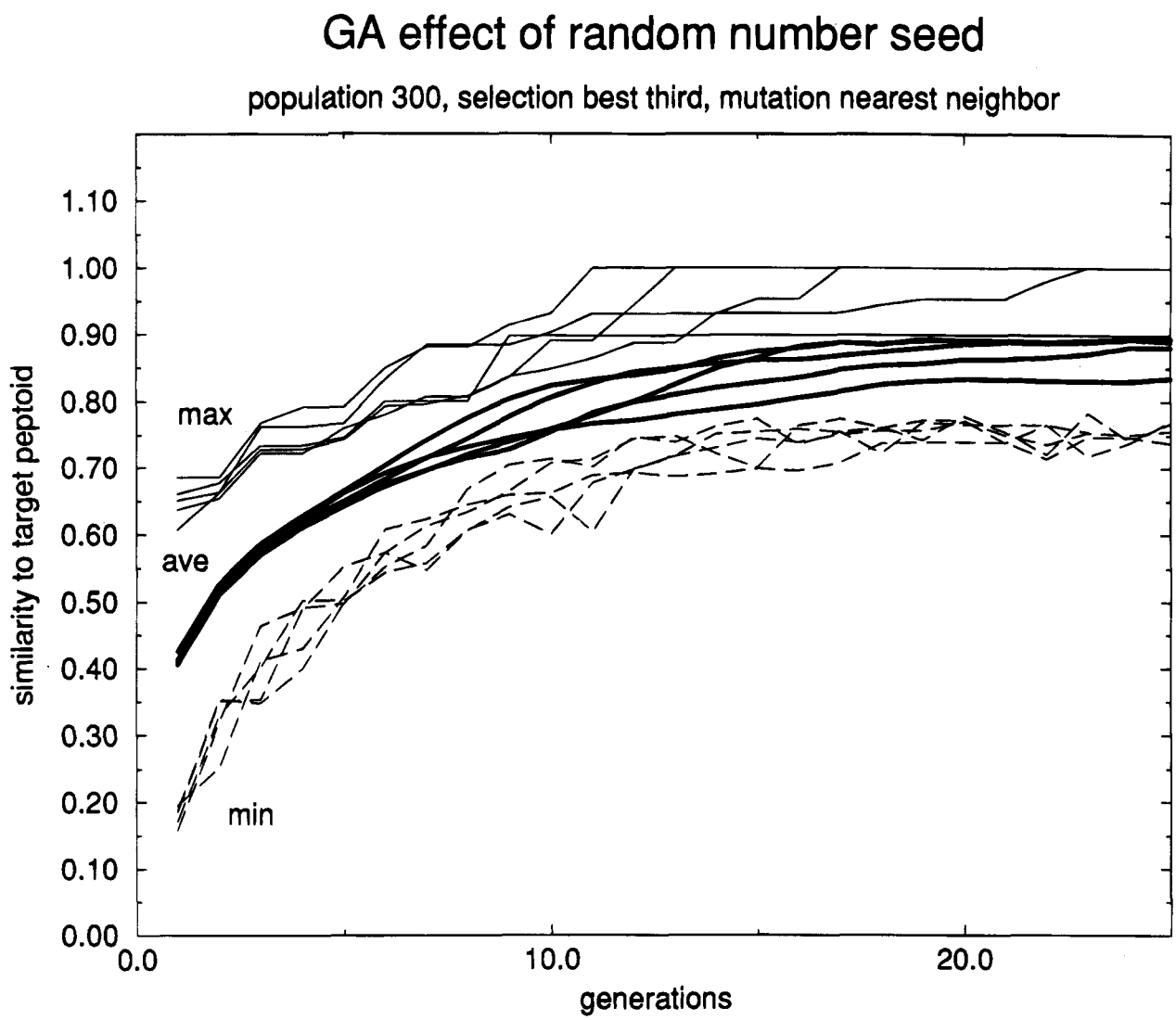


Figure 4. A graph of the minimum, average, and maximum score as a function of generation for a population of 300 against a target peptoid. Here the best third selection method and the nearest neighbor mutation method are used, but five different random number seeds are used at the beginning of each run.

Statistics for the three runs are shown in Table 1. There are many possible ways to calculate the "merit" of a fragment, i.e., its association with high-scoring compounds. One straightforward way is to simply look at the frequency of the fragments in the population at the end of the run. (Looking at just the elite third of the population does not change the relative ranking of fragments.) The frequency of the most common residues, combined over the three runs, is shown in Figure 6. There is a reasonable resemblance between the tripeptoids with these fragments and the probes. Clearly, residue F corresponds to the N-terminus of the probes; a particular indole-containing fragment (205) is strongly preferred. Residue D prefers a variety of fragments containing a carboxylate and aliphatic group similar to the middle two residues of the probe. Residue B prefers phenyl-containing fragments similar to the C-terminus of the probe. What is not visible from this figure is that, although the final scores are nearly equal, different runs generate different subsets of fragments, i.e., tend to select different local maxima in combinatorial space. For instance, fragments 904 and 1970 are selected in the first two runs, but not in the third. Fragment 100 is found only in the third run, although at least one topologically equivalent fragment is found in all runs. One can find many more examples. This once again underscores the importance of doing at least a few runs with different starting random number seeds.

ACE Trend Vector. Our third example demonstrates the use of a trend vector to score the tripeptoids. Most structures in MDDR have associated text fields, including "activity category". We found in the MDDR database 223 compounds that refer to "angiotensinase inhibitor" (ACE) in the activity category field. These we assigned an activity of 1.0. We randomly selected about 2000 MDDR compounds for which this activity is not mentioned and assigned these an activity of 0.0. So that large classes of closely related inhibitors would not dominate the trend vector analysis, we eliminated compounds within the set so that no two compounds had a similarity greater than 0.6. The final training set had 1172 compounds, of which 35 were actives. Typical actives are shown at the top of Figure 7. The trend vector calculated from this training set had five highly significant partial-least-squares components (at 21, 19, 7, 8, and 5 standard deviations above chance), and the correlation between predicted and observed activity in the fit was 0.82. The trend vector summarizes the chemical features shared by ACE inhibitors that differ from the general population of druglike molecules. We did three runs with the same protocol as before, except that the ACE trend vector was used for scoring.

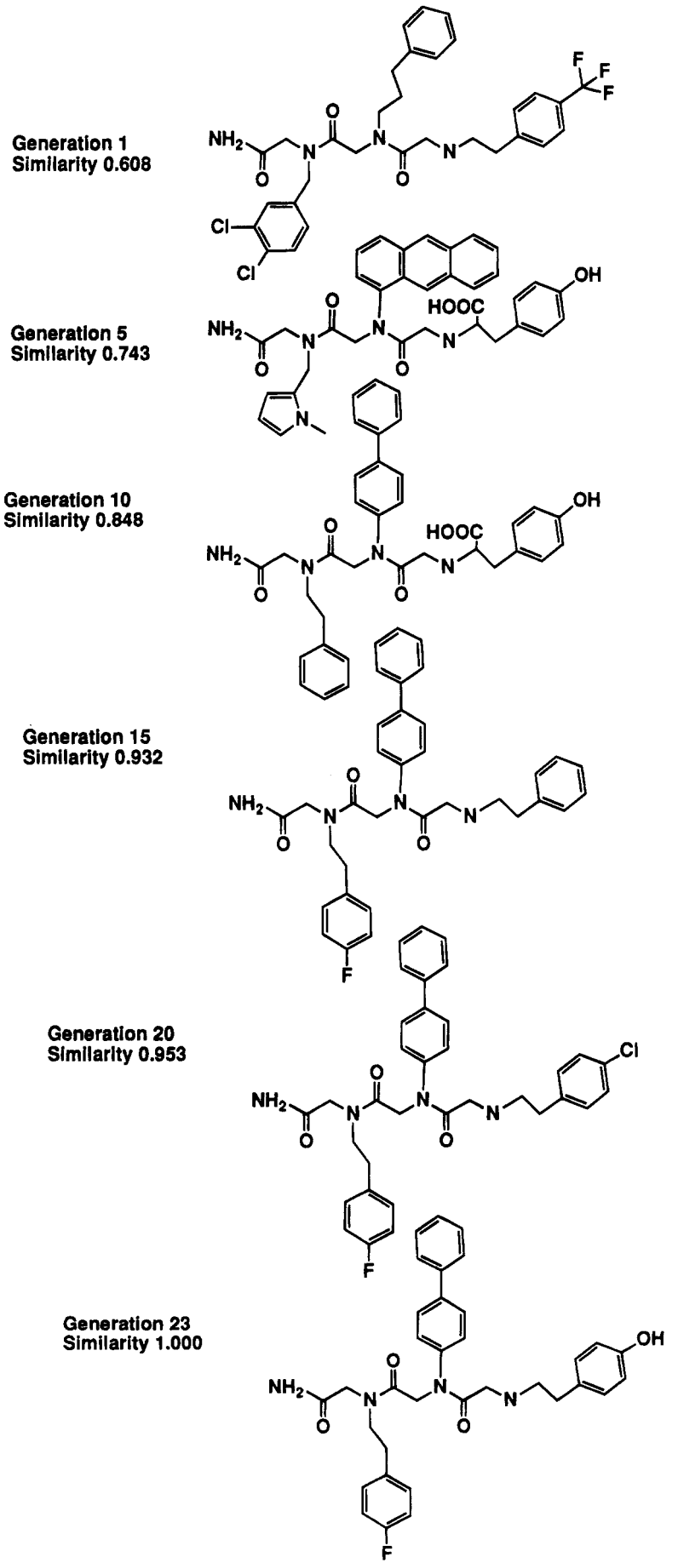


Figure 5. The highest-scoring compound as a function of generation. This is from the run in Figure 4 in which the maximum score reached 1.0 in 23 generations.

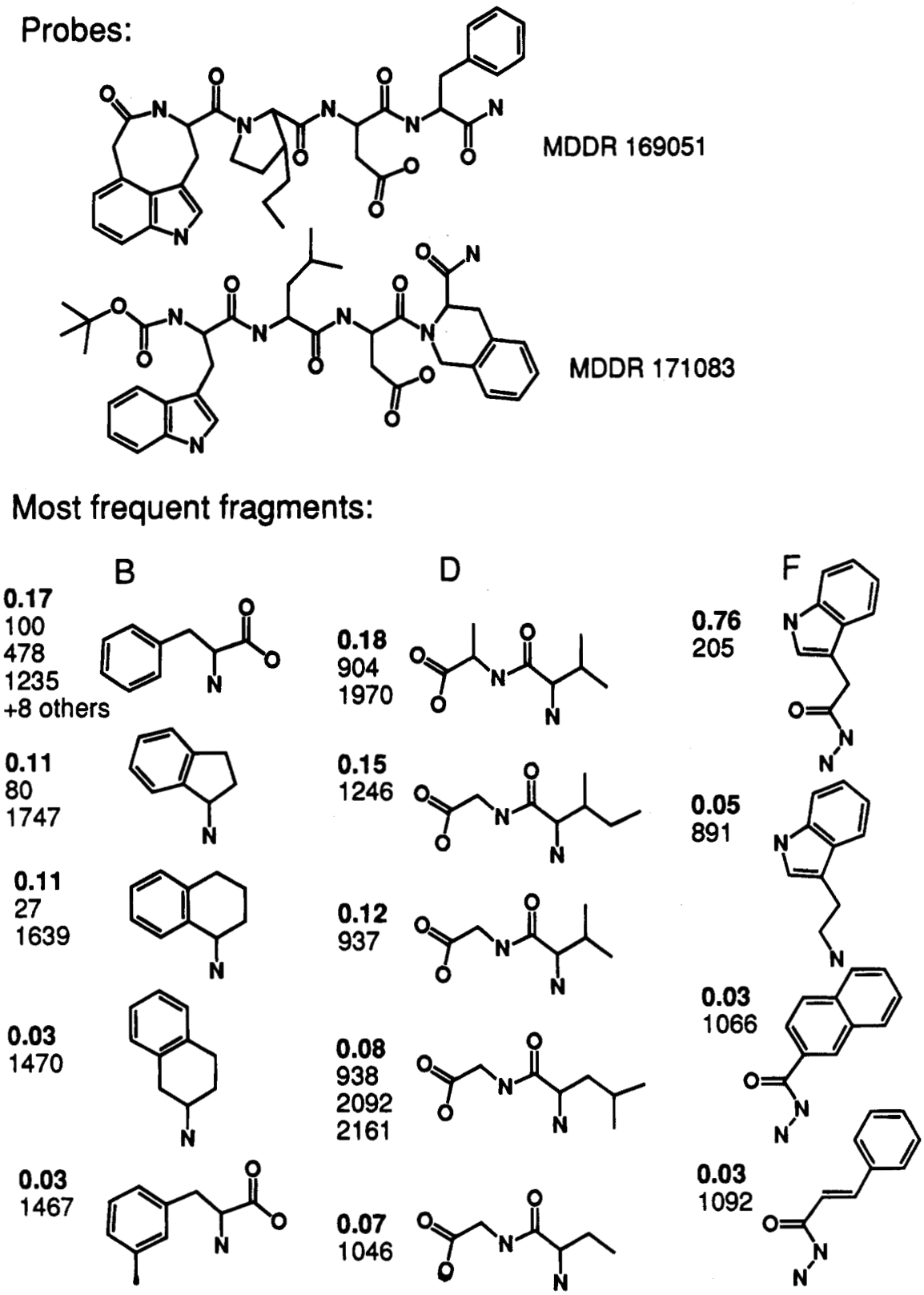


Figure 6. The example with tetrapeptide CCK antagonists as similarity probes. (above) structures of the probes and (below) the most frequently occurring fragments in generation 25 averaged over three runs. Topologically equivalent fragments have been combined. Associated with each fragment structure is its frequency (in bold) and the list of fragments with that structure.

Statistics for the three runs are shown in Table 2. The most frequent fragments at the end of the run are shown in Figure 7. Not visible from the figure is the correlation between fragments. Residue F strongly prefers the Ala-Pro dipeptide (fragments 1635 and 2697), and residue B prefers aromatic rings at the end of short alkyl chains (e.g., fragment 564), as found in many of the actives. Residue D prefers carboxylates and phosphates. Carboxylates, esters, and phosphoesters are observed in many ACE inhibitors. We also see the reversed situation where the aromatic group (fragments 2587 and 2579) are at residue F, and fragments like 540 and 545 are at residue B. Apparently, the trend vector does not prefer a particular direction in assembling the tripeptoids. Again we see that there is a difference

Table 1. Statistics for Three Runs Using Similarity to Two Tetrapeptide CCK Antagonists as the Score

	generation 1				generation 25				generations
run	n	av[a]	min.	max.	*n*	av	min.	max.	at converg[b]
1	300	0.44 ± 0.07	0.23	0.62	300	0.68 ± 0.03	0.61	0.72	21
2	300	0.43 ± 0.07	0.26	0.57	300	0.68 ± 0.02	0.60	0.72	17
3	300	0.42 ± 0.07	0.21	0.60	300	0.69 ± 0.02	0.62	0.72	13

[a] Mean ± one standard deviation. [b] Generation at which the maximum score reaches its limiting value.

Table 2. Statistics for Three Runs Using Predicted Activity on the ACE Trend Vector as the Score

	generation 1				generation 25				generations
run	*n*[a]	av[b]	min.	max.	*n*	av[b]	min.	max.	at converg
1	281	0.53 ± 0.01	0.50	0.57	297	0.60 ± 0.01	0.57	0.61	11
2	283	0.53 ± 0.01	0.50	0.57	298	0.60 ± 0.01	0.56	0.61	11
3	277	0.53 ± 0.01	0.50	0.57	299	0.60 ± 0.01	0.57	0.62	12

[a] We count only the molecules for which there was a trend vector prediction (score >0). Hence $n < 300$. [b] These values can be compared to the average score of the actives in the ACE training set: 0.60 ± 0.02.

between runs in what fragments are found. For instance, the first run exclusively favored fragment 2697 for residue F, while the second run exclusively favored fragment 1635. The reverse molecules were observed only in the third run.

DISCUSSION

We have demonstrated that one can use a genetic algorithm to find the structures of molecules that score maximally on a topological SAR, given a particular form of a combinatorial library (e.g., tripeptoids) and a particular universe of fragments to choose from (e.g., suitable amines from the FCD). This can be accomplished within a modest amount of computer time, even when the universe of fragments is large. At the end of the genetic algorithm run one has a list of fragments that occur in the maximally-scoring molecules. A basis set of a combinatorial library targeted at the biological activity for which the SAR was derived should probably contain at least some of these fragments. It should be emphasized that the goal here is to design a library with the maximal chance of containing actives in one particular biological assay, and so the basis set might include many similar fragments. This is in contrast to the more usual goal of designing a combinatorial library to be tested against a set of different biological assays, in which case one wants a basis set with maximally dissimilar fragments to cover the most chemical space.

Further discussion will be on four aspects: antecedents for this work, future work on the genetic algorithm protocols, general comments on molecular assembly, comments on the scoring methods, and a caveat on the general usefulness of this approach.

The idea of stochastically constructing molecules from fragments and scoring them against a similarity probe or trend vector was first proposed by Nilakantan et al.[14] as a way of suggesting novel chemical structures for synthesis. One critical difference between that work and this is the nature of the fragments. Their fragments were rings, acyclic groups, and functional groups parsed from druglike compounds. These fragments could be joined at any open valence. The fact that the final molecules were seldom synthesizable was an important drawback. Here our fragments are chemically convenient modules, and they may be joined only at specified atoms, consistent with a particular scheme for combinatorial synthesis. Thus synthesizability is assured. Another difference is that Nilakantan et al. did a single selection from a large population, whereas the genetic algorithm does a series of selections starting from a smaller population. The latter scheme has a much better chance of eventually finding higher-scoring structures.

The literature on genetic algorithms details many diverse protocols for choosing population size, for making selections, mutations, and crossovers, and protocols for fixing the relative frequency of the three. Many of these protocols are aimed at reaching the best compromise between finding good answers quickly and finding a large variety of answers. Our protocols have been very simple, and we have not tested many different ones. Thus we cannot claim that the particular protocol given here is optimal, only that it appears adequate, at least for the case where there are few residues with many possibilities at each residue. There is probably some room for improvement.

Since the goal of this exercise is not only to find the highest-scoring molecules, but to select a subset of interesting fragments, another consideration is how to best select the fragments. Here we looked at the frequencies of fragments at the end of the run as one figure of merit. Clearly there many other possibilities: taking the union of fragments regardless of their frequencies, looking at the frequencies just after convergence instead of at the end, etc. This is an area of active investigation.

We have shown peptoids as examples, but the assembly method can apply to any type of molecule where there are well-defined chemical modules and where a set of fusion instructions can be written. The molecules need not even be linear. For instance benzodiazepines, another target for combinatorial synthesis,[2] have a central core with several substituents. In this case, one would start with the benzodiazepine ring and add the substituents in any order.

Our demonstration of the genetic algorithm was based on scoring by two particular topological methods, similarity probes and trend vectors. One can think of a number of variations. For instance, the fitness score need not be identical with the similarity or predicted activity; the score could be any function of these values. Also, in the trend vector case molecules for which predictions could not be made were here given a score of zero. An alternative way would be to penalize the score by how much the descriptor

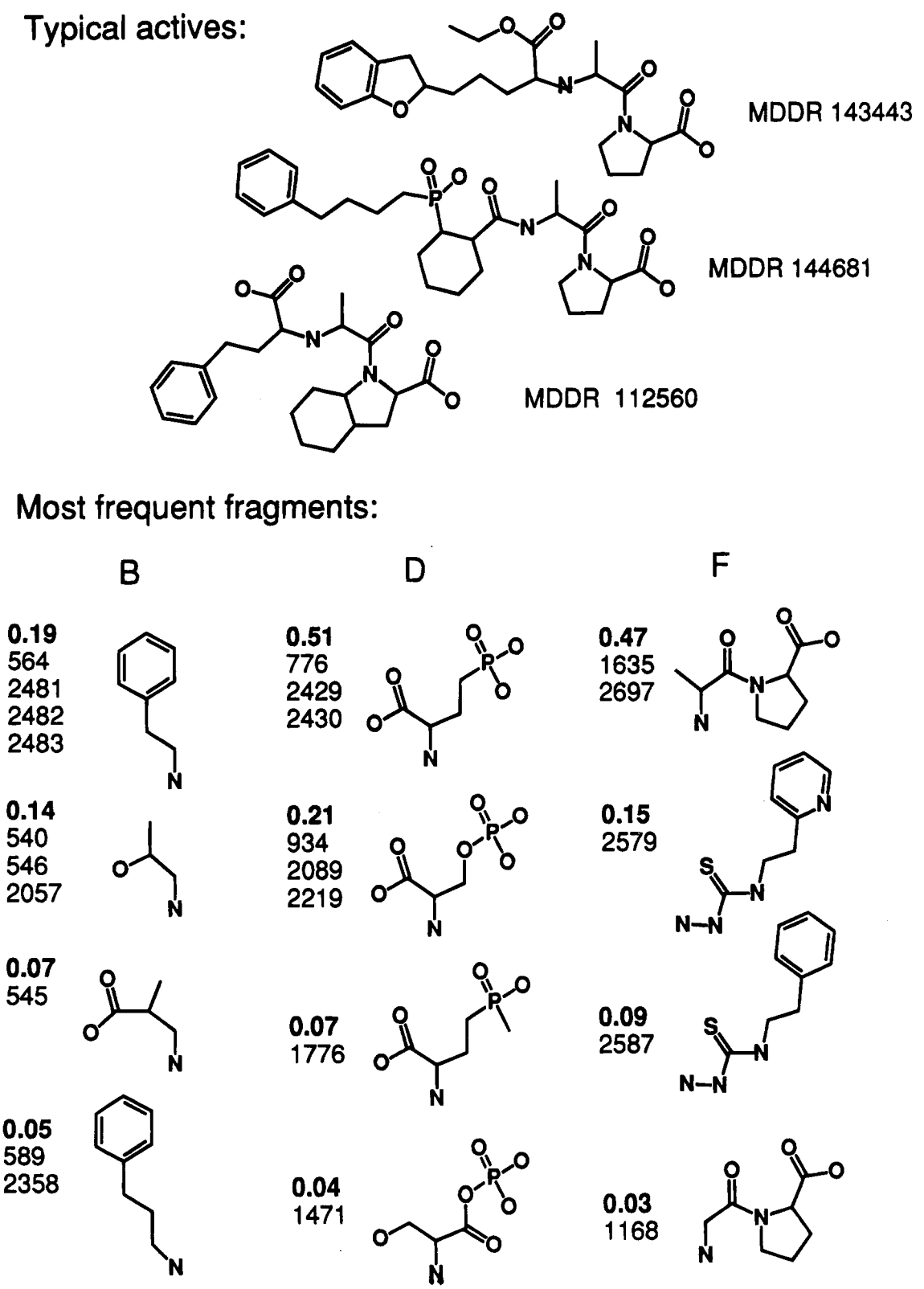


Figure 7. The example of scoring by the ACE trend vector: (above) typical angiotensinase inhibitors and (below) the most frequently occurring fragments in generation 25 averaged over three runs.

fraction violated the 95% cutoff. That way potentially good molecules would survive longer.

One useful feature of genetic algorithms is that they can be used with any type of scoring. Topological methods have the advantage that calculating the score is very fast, but there are limits to a topological view of chemistry, especially in that conformational flexibility and stereochemistry are ignored. One can easily imagine a type of scoring based on 3D properties, although scoring on a 3D basis is likely to be more computationally demanding.

An important caveat about any type of SAR scoring is that, for this method of fragment selection to work in a practical sense, not only does the genetic algorithm have to be a good optimizer but also the SAR method has to predict which molecules will have the proper activity. It is therefore imperative that the SAR method be applicable across

chemical classes. Similarity probes and trend vectors using the atom pair descriptor have in the past been very successful in predicting activity across chemical classes of druglike molecules. However, it is not known whether predictions generated from druglike molecules will apply to specific oligomers, e.g., peptoids, and it is not known whether the oligomer backbone is compatible with the particular activity. This can only be settled empirically.

ACKNOWLEDGMENT

We thank Dr. Robert B. Nachbar for useful discussions on genetic algorithms. We also thank applications molecular modelers and the medicinal chemists at Merck involved in combinatorial synthesis, especially Dr. Kevin Chapman, for the opportunities to test this method. Dr. Michael D. Miller devised the random-access database for storing the fragment libraries.

REFERENCES AND NOTES

(1) Gallop, M. A.; Barret, R. W.; Dower, W. J.; Fodor, S. P. A.; Gordon, E. M. Applications of combinatorial technologies to drug discovery. 1. Background and peptide combinatorial libraries. *J. Med. Chem.* **1994**, *37*, 1233−1251.

(2) Gordon, E. M.; Barret, R. W.; Dower, W. J.; Fodor, S. P. A.; Gallop, M. A. Applications of combinatorial technologies to drug discovery. 2. Combinatorial organic synthesis, library screening strategies, and future directions. *J. Med. Chem.* **1994**, *37*, 1385−1401.

(3) Simon, R. J.; Kania, R. S.; Zuckermann, R. N.; Huebner, V. D.; Jewell, D. A.; Banville, S.; Ng, S.; Wang, L.; Rosenberg, S.; Marlow, C. K.; Spellmeyer, D. C.; Tan, R.; Frankel, A. D.; Santi, D. V.; Cohen, F. E.; Bartlett, P. A. Peptoids: a modular approach to drug discovery. *Proc. Natl. Acad. Sci. U.S.A.* **1992**, *89*, 9367−9371.

(4) Zuckermann, R. N.; Kerr, J. M.; Kent, S. B. H.; Moos, W. H. Efficient method for the preparation of peptoids (oligo N-substituted glycines) by submonomer solid-phase synthesis. *J. Am. Chem. Soc.* **1992**, *114*,10646−10647.

(5) Zuckermann, R. N.; Martin, E. J.; Spellmeyer, D. C.; Stauber, G. B.; Shoemaker, K. R.; Kerr, J. M.; Figliozzi, G. M.; Goff, D. A.; Siani, M. A.; Simon, R. J.; Banville, S. C.; Brown, E. G.; Wang, L.; Richter, L. S.; Moos, W. H. Discovery of nanomolar ligands for 7-transmembrane G-protein-coupled receptors from a diverse N-(substituted)-glycine peptoid library. *J. Med. Chem.* **1994**, *37*, 2678−2685.

(6) The Fine Chemicals Directory is distributed by Molecular Design Ltd., San Leandro, CA, 1993.

(7) Carhart, R. E.; Smith, D. H.; Venkataraghavan, R. Atom pairs as molecular features in structure−activity studies: definition and applications. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 64−73.

(8) Sheridan, R. P.; Nachbar, R. B.; Bush, B. L. Extending the trend vector: the trend matrix and sample-based partial least squares. *J. Comput.-Aided Molec. Design* **1994**, *8*, 323−340.

(9) Sheridan, R. P.; Venkataraghavan, R. New methods in computer-aided drug design. *Acc. Chem. Res.* **1987**, *20*, 322−329.

(10) Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning;* Addison-Wesley: New York, 1989.

(11) *Handbook of Genetic Algorithms*; Davis, L., Ed.; van Nostrand Reinhold: New York, 1991.

(12) Nilakantan, R.; Bauman, N.; Dixon, J. S.; Venkataraghavan, R. Topological torsion: a new molecular descriptor for SAR applications. Comparison with other descriptors. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 82−85.

(13) MACCS-II Drug Data Report (V 93.1) is distributed by Molecular Design Ltd., San Leandro, CA, 1993.

(14) Nilakantan, R.; Bauman, N.; Venkataraghavan, R. A method for automatic generation of novel chemical structures and its potential applications to drug discovery. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 527−530.

CI940129Q