Articles

Designing Combinatorial Library Mixtures Using a Genetic Algorithm

Robert D. Brown* and Yvonne C. Martin

Pharmaceutical Products Division, Abbott Laboratories, D47E/AP10, 100 Abbott Park Road, Abbott Park, Illinois 60064-3500

Received January 15, 1997[⊗]

The design of combinatorial mixture libraries should take account of a number of factors. This paper describes the application of a genetic algorithm to optimizing the diversity of libraries while minimizing the effort that will be needed to deconvolute the biological hits by mass-spectroscopic techniques. It differs from previous applications of genetic algorithms to combinatorial library design in that each chromosome encodes an entire library with the result that properties of the library are optimized. Our method is such that it is easily extensible to optimizing the distributions of any number of physical or other properties of the library. The method allows for the combinatorial constraint inherent in mixtures that every substituent at each diversity site must occur in combination with every substituent at every other site. We present results showing that the genetic algorithm can produce good library designs in a timely manner.

Introduction

When designing a combinatorial mixture library for lead identification, it is desirable that the compounds within that library be as diverse as possible, to fully explore the scope of activity against the target. However the design of a library should take into account many other factors, not least among them the effort needed to deconvolute the mixtures once hits are obtained. Factors such as the cost and availability of reagents, for example, or the ranges of physical properties of the library products may also require optimization. Furthermore, in the design of a mixture the combinatorial constraint always applies, *i.e.*, every substituent at each position will occur in combination with every substituent at all other positions. Each of these additional considerations may mean that a certain amount of possible diversity in a library has to be sacrificed.

In this paper we present a method that specifically addresses the trade-offs involved in designing a library for both optimum diversity and efficient deconvolution. However, this method also provides a framework into which any number of additional factors may easily be incorporated. The method is specifically aimed at mixtures produced using methods such as mix-and-split or competitive coupling in which the combinatorial constraint applies. For other synthesis protocols such as parallel synthesis, this constraint does not apply. Individual compounds can be selected for synthesis in these cases, and so other design methods may be appropriate.

The Mixture Library Design Problem

Library Diversity. Several methods have been proposed to quantify diversity.[1−7] One approach to producing diverse library products has been to ensure that the sets of precursors used to construct the library are as diverse as possible.[4,8] If all suitable precursors for a given diversity site are grouped on the basis of some desirable features, such as distribution of potential pharmacophore points, then selecting no more than one from each group should ensure that the set is diverse. To produce a diverse library it is therefore desirable to ensure that every precursor is taken from a different group.

A recent study has suggested that maximizing diversity among the precursor sets may not necessarily give the most diverse possible set of library products.[9] An alternative is to consider the diversity among the library products themselves. This may be measured by enumerating the library products and either clustering them and attempting to pick as few compounds as possible from each cluster[10] or using a cell-based partitioning method[11−13] and attempting to pick as few compounds as possible from each cell.

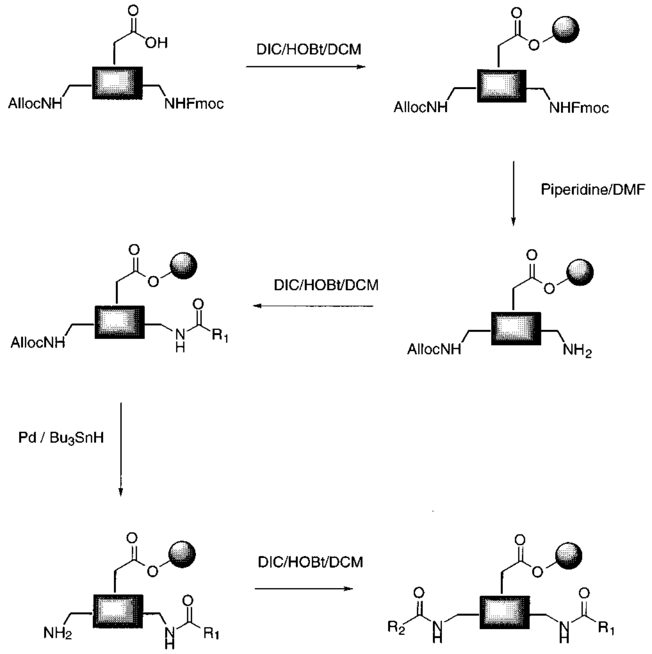
Deconvolution. A number of solutions have been suggested to the problem of deconvoluting the hits from a mixture library. These include tagging beads with various types of chemically or spectroscopically readable labels or producing libraries on silicon chips whose identities can later be determined by radiofrequency scanning.[14] Our method is designed to address the problem of deconvolution when using mass-spectroscopic techniques to identify the molecular ion of active compounds.[15] Once this information is known, selective resynthesis and testing of some or all of the compounds in the library with those molecular weights will be required. Since it is desirable to keep this work to a minimum, it is sensible to design a combinatorial library to have the smallest possible number of compounds of any one molecular weight, given the constraints of the required size of the library and the availability of suitable precursors.

For a given number of library products all having the same molecular weight, deconvolution will be simplified further if the substituents used at the diversity sites have different molecular weights. It is therefore desirable to also minimize the substituent molecular weight redundancies, for each given product molecular weight.

In a mix-and-split strategy for combinatorial synthesis,[16] the pools of compounds that result from the addition of the final diversity site are often not mixed. Since these pools are screened individually, the decon-

[⊗] Abstract published in *Advance ACS Abstracts,* June 15, 1997.

Scheme 1. Reaction Scheme for the Synthesis of Example Library I



volution problem applies to each subpool separately; however, the diversity optimization applies across the library as a whole. A library design strategy must therefore be able to suggest pools within each of which there is the minimum product and substituent molecular weight redundancy. At the same time the diversity of the library should be optimized over all the pools.

It is important to note that the design requires the optimization of the whole library, not selection of individual library compounds, and thus requires the consideration, and comparison, of all possible libraries rather than all possible library compounds. In a combinatorial design it is, in any case, impossible to select individual combinations of precursors without making all other products containing those precursors.

Problem Size. The number of precursors that are suitable for use in constructing a combinatorial mixture library is often larger than the number that can reasonably be used. Hence, typical mixture library design problems have huge search spaces, the sizes of which are best illustrated by two of the examples to which we have applied our method. The design of these libraries is discussed in the results section of this paper.

In library I, two sites of diversity were available. A reaction scheme for the production of this library is given in Scheme 1. There were 360 commercially available precursors compatible with the chemistry for R_1 and 259 for R_2. There were therefore 93 240 possible library product compounds. The design of this library called for the production of 10 000 compounds by combining 100 R_1s with 100 R_2s.

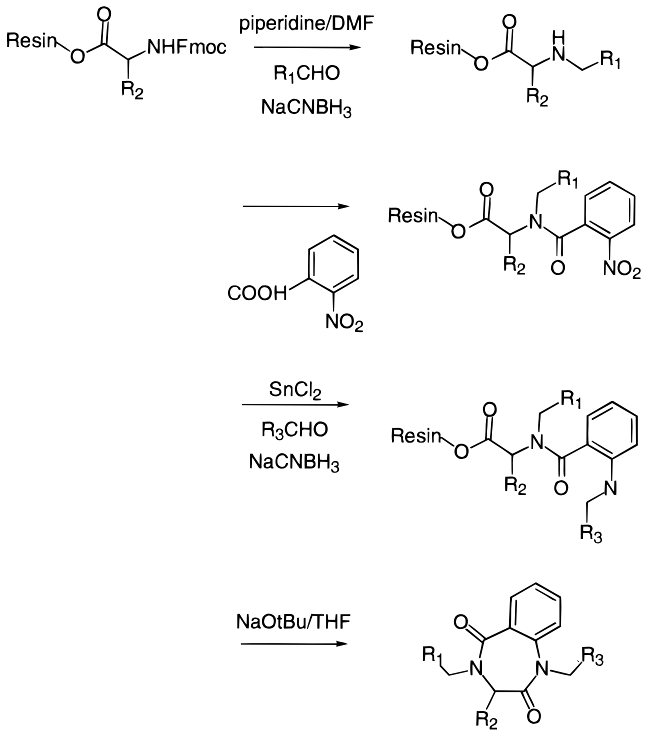
The number of ways of selecting *k* objects from *n* is

$$^{n}C_k = \frac{n!}{(n-k)!k!}$$

and so the number of possible libraries is

$$^{360}C_{100} \cdot {}^{259}C_{100} = 2.5 \times 10^{164}$$

Scheme 2. Proposed Reaction Scheme for the Synthesis of Example Library II[a]



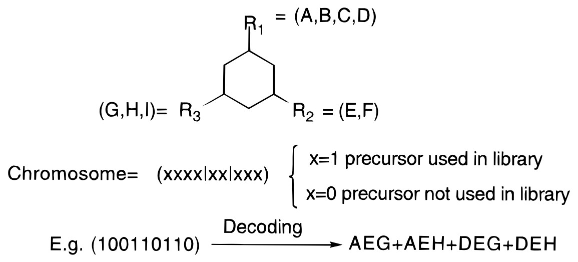
[a] This is partly based on the scheme of Mayer *et al.*[32]

To put this number into perspective, the age of the universe is approximately 10^{17} s and its size 10^{108} Å.[3]

Library II, for which a reaction scheme is shown in Scheme 2, had three sites of diversity on a small molecule core. In the existing inventory of precursors available within Abbott at the time at which the library was to be constructed, there were 53 suitable candidates for R_1 and R_3 and 42 for R_2. No fixed size was set for this library. Instead, a maximum allowed redundancy at any one molecular weight was specified and the largest possible size library required. In this case therefore, all combinations of picking between 1 and 53 compounds for R_1, while picking between 1 and 42 for R_2 and 1 and 53 for R_3, are allowed, giving approximately 10^{194} possible libraries each containing between 1 and 117 978 compounds.

The problem size makes it impossible to tackle by enumeration. For example, if 100 potential libraries could be evaluated for their mass redundancy and diversity in one CPU second, an exhaustive enumeration of all possible solutions for library I would required on the order of 10^{154} years.

In summary, library design is a complex problem, requiring the optimization of a number of often competing factors, over a vast search space. Genetic algorithms have been successfully applied to a wide range of such problems in both chemical and nonchemical domains.[17] A genetic algorithm is a computational technique that mimics the processes of Darwinian evolution.[17−19] A potential solution to a problem is encoded in a representation termed a *chromosome*. This is typically a string of bits, integers, real numbers, or symbols each of which is termed a *gene*. The genetic algorithm operates on a *population* of these chromosomes that are generated by assigning values to the genes in the chromosomes, often at random. A *fitness function*



R_1 = (A,B,C,D)
(G,H,I)= R_3
R_2 = (E,F)
Chromosome= (xxxx|xx|xxx)
x=1 precursor used in library
x=0 precursor not used in library
Decoding
E.g. (100110110)
AEG+AEH+DEG+DEH

Figure 1. Chromosome encoding. The chromosome encodes a representation of a library in a bit string. Each gene (bit) corresponds to one precursor and is set to 1 if that precursor is to be used in that library. Knowing the positions of the R group boundaries allows the chromosome to be decoded to give the library product combinations as shown.

measures how well adapted each chromosome is to its environment. In our example this would equate to how diverse the library represented by the chromosome is and how easy it would be to deconvolute. Once an initial *parent* population is generated, it is subjected to some evolutionary processes to breed a new *child* population, with the better adapted parents being allowed a greater chance to produce children. As with natural evolution, over successive generations the members of the population become better adapted to their environment, and thus better solutions to the problem in hand are discovered.

There are two prerequisites to being able to apply a genetic algorithm to a problem. The first is to be able to choose a representation that allows every possible solution to the problem to be encoded in a chromosome. The second is that it must be possible to write a fitness function to decode the chromosome and produce a score that reflects the quality of that solution.

GALOPED - *G*enetic *A*lgorithm for *L*ibrary *Op*timization for *E*fficient *D*econvolution

We have developed a program, GALOPED, that applies a genetic algorithm to the library design problem. The program is implemented in C for Silicon Graphics UNIX workstations using the SUGAL package.[20] SUGAL implements a wide range of different methods for each process in a genetic algorithm and provides a mechanism for these to be optimized in combinations. It requires the user to provide C code for the fitness function, data entry, and results output and for any nonstandard genetic algorithm operators to be used. The package includes a MOTIF interface to allow parameter setting and monitoring of the progress of the program. The interface was extended to create windows for browsing the frequency distributions and the chemical structures of the precursor sets for each chromosome in a population.

Encoding Strategy. In our program each chromosome represents a different potential library. Each is therefore nominally split into a number of sections corresponding to the number of diversity sites. The number of genes in each section is equivalent to the number of available precursors for that position. The genes are binary, a 1 indicating that the precursor is to be used at that diversity site and a 0 that it is not. A gene is said to be set if its value is 1 and unset if it is 0. A simple example of this representation is shown in Figure 1, in which there are three sites of diversity, with R_1 having four suitable precursors, A−D; R_2 having two, E and F; and R_3 having three: G−I. This requires a 9-bit chromosome with the first 4 bits representing the first diversity site and specifically bit 1 representing the presence or absence of precursor A in the library represented by that chromosome. It is important to note that given only the gene positions of the section boundaries, it is possible to decode a chromosome to give the precursor combinations in the products that are required by the fitness function.

Substituents that are specified to be included in all solutions are not allocated a gene position in the chromosome, since the chromosome represents choices that can be made. However they are included when enumerating the substituent combinations of the library products during the fitness function evaluation.

In some library syntheses, a set of precursors is added to more than one position simultaneously, and therefore the same precursor set must be used at each of these positions. The program allows one or more diversity positions to be fixed as equal to another. In this case the repeated precursor set(s) is(are) not represented in the chromosome but included in the enumeration of products during the fitness function evaluation.

It should be noted that this encoding is different from other genetic algorithm applications to combinatorial chemistry which have been reported recently.[21−23] In those studies a single library product is encoded in each chromosome and a population of individual library products is optimized rather than a population of complete libraries.

A step-by-step procedure for the GALOPED program is as follows:

1. Input substituent sets, diversity measurements, and design criteria.
2. Generate an initial population of chromosomes by random initialization.
3. Calculate the fitness for each chromosome.
4. Create the mating pool by biased random selection of parents.
5. Create children by crossover and mutation of two randomly selected parents from the mating pool.
6. Calculate the fitness of the children.
7. Insert (fitter) children into population by displacing (weaker) parents.
8. If the maximum number of generations has not been reached and there have been improvements in fitness, the last *n* generations go to step 3.
9. Display results.

Each stage of the above procedure will now be described in more detail.

Step 1. Program Input. When different structural classes of precursor are being used at the same diversity site (for example, both carboxylic acids and acid chlorides might be compatible with the chemistry at a given R position), it is necessary to transform the precursors into *substituents* so that the correct molecular weights can be calculated and correct pharmacophore point types assigned. A substituent is the form in which a precursor will occur in the library products. A number of programs may be used to achieve this transformation.[24−26]

After producing the substituent lists, the necessary diversity calculations must be performed. If diversity of the precursor sets is to be considered, the substituents to be used at each diversity site are separately grouped into 2D clusters or 3D families. We do not currently combine 2D and 3D descriptors.

2D clusters are produced using MACCS structural keys followed by Ward's agglomerative clustering.[10] Ward's clustering makes use of pairwise Euclidean distances calculated from the MACCS keys. We have conducted extensive validation studies on these methods, looking at their ability to separate known active and inactive structures in a number of datasets. We have shown the methods to be able to achieve good separations of structures into biohomogeneous clusters, *i.e.*, those containing mostly structures of one activity class.[10] Others have reached the same conclusions studying the same types of descriptor.[27]

3D families are produced by identifying all potential pharmacophore points in each precursor and then grouping together those having identical patterns of all pharmacophore points, within a given distance tolerance, in the single conformation produced by the CONCORD program.[8,28] This is a partitioning procedure, rather than clustering, since every compound with a particular pharmacophore pattern is assigned to the same family.

These cluster or family numbers are used by the fitness function to assess the diversity of the subsets of precursors selected for a particular library, since each 2D cluster or 3D family contains structures with a common set of features. Our validation studies have suggested that by selecting one from each cluster or family, there is a good chance that all activity classes present in the dataset as a whole will be represented in the selected set.

If diversity is to be assessed over the product structures, then all possible product structures are enumerated. For all libraries of nontrivial size the 3D grouping procedure is too slow to be used. Therefore, clusters are produced using MACCS keys and Ward's agglomerative clustering. Due to the time requirements of the clustering algorithm, there is a practical limit of around 200 000 library products using this method. As an alternative we are currently investigating the use of cell membership numbers produced using the Diverse-Solutions software[11] which, initial experiments indicate, should allow sets of several million structures to be processed.

Input to the program consists of a list of named substituents with structures in SMILES format, each with an indication of its R position, group number, and whether it must be included in all suggested solutions. The substituent structures are necessary to allow browsing of the chosen sets once the design is finished. Properties of the products, such as cluster or cell numbers, are indexed using composite names formed from concatenation of the precursor names to allow their lookup by the fitness function. A number of other parameters that must be defined by the user are discussed in subsequent sections.

Step 2. Initializing the Population. A fixed size library design implies that a fixed number of genes must be set in each section of the chromosome in the final solution. To allow this, user-defined upper and lower bounds may be set on the number of genes within each section. A chromosome generated by any genetic operator that breaks these bounds is disallowed. Once the lower bound is set to the required number of substituents, the fitness function has the effect of driving the chromosomes toward solutions containing this number of genes per section, since the simplest way to relieve excess redundancies is to use fewer substituents. Chromosomes with more than the minimum number of bits but less than the upper bound are permitted in the population to allow the operators to produce more valid chromosomes than would otherwise be possible and to allow more diversity among the solutions to be maintained while the population is being optimized.

For library designs with no fixed sizes, the upper and lower bounds are useful to reduce the search space. This they do by preventing consideration of solutions that greatly exceed the frequency cutoffs or have so few substituents as to be uninteresting. In this case the requirement to include as many bits as possible prevents the fitness function from driving the number of genes toward the lower bound.

Each chromosome may be initialized by picking a different random number of genes between the upper and lower bounds to be set in each section of each chromosome and then randomly selecting gene positions until that number of genes has been set.

A reviewer of this paper suggested that there may be an advantage to biasing the initialization toward diverse substituents, in cases in which the diversity of the substituents rather than the products is being optimized. This has been implemented by selecting a random number of genes between the upper and lower bounds, as before. Gene positions are again selected at random to be set, but this time a gene position is disallowed if another representing a substituent from the same cluster or family has already been set. If this is the case, it is left unset and the next random choice examined. A population size of around 100 is typically used.

Step 3. Fitness Function. The fitness function consists of a number of parts, each of which optimizes a different aspect of the library: the molecular weight/formulae redundancy, the substituent molecular weight redundancies, the number of compounds, and the diversity.

The method of evaluating the redundancy of the library is dependent on the type of mass spectroscopy to be used in the assay stage. For low-resolution MS, molecular weights are treated as integral. For a high-resolution experiment, it is considered sufficient that the molecular formulae of the products not be redundant, so the number of occurrences of each unique formula is counted in this case.

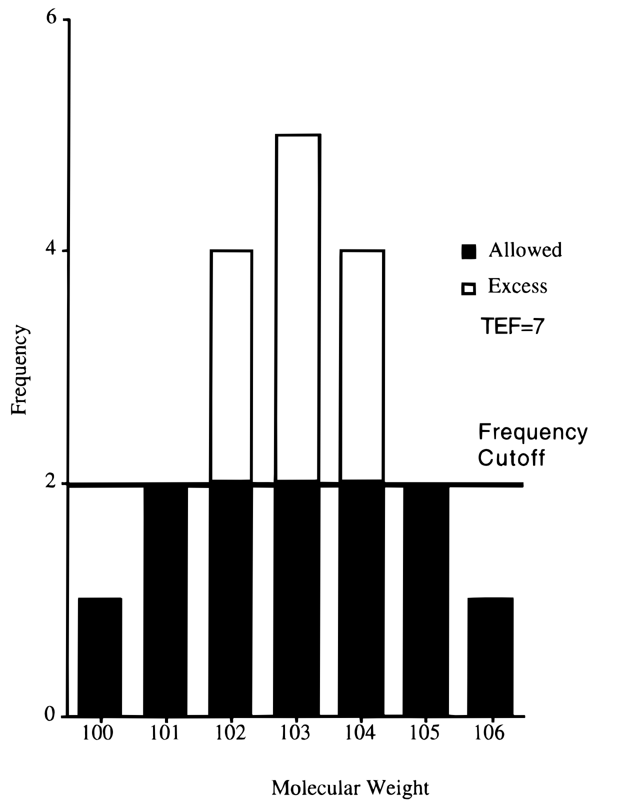


Figure 2. Calculation of excess frequency of molecular weight for scoring a chromosome. A frequency distribution of the number of occurrences of library products with each molecular weight is calculated. The total number of occurrences over a user-defined cutoff is calculated across the distribution.

To evaluate the redundancy among molecular weights (or molecular formulae), the chromosome must first be decoded to identify the library products. The library products encoded in a chromosome may be enumerated by taking the substituents corresponding to every set gene in the chromosome section for R_1 in combination with every set gene in R_2 in combination with every set gene in R_3, etc. If the library synthesis strategy is to not mix the final position, then all combinations are taken over the first $N - 1$ sections rather than all N sections, since it is the frequency redundancies within each pool that are of concern. The molecular weight or formula for each substituent is calculated during the initial data loading so that the products may be quickly calculated by summing those of the relevant substituents; a constant may be added to take account of a core, although this is not necessary for the score. A frequency count of each unique weight or formula is kept over all the library products. Note that is it not necessary to construct the actual structures of the library products but rather simply to sum together the weights or formulae of the substituents.

To produce a redundancy score for the library, the total number of molecular weight redundancies over a user-defined cutoff is summed across the distribution, to give the *excess frequency* (see Figure 2). After some experimentation it was decided to use the squares of the individual excess frequencies at each molecular weight/formula. This tends to flatten the whole frequency distribution by ensuring, for example, that a distribution in which two separate molecular weights have an occurrence one greater than the cutoff is more favorable than one in which one weight has an excess frequency of two and a second of zero.

A score for the substituent molecular weight redundancies is produced in a similar fashion. For each product molecular weight a separate frequency distribution is calculated for all the substituents in the library compounds having that molecular weight. A separate score for each of these, again based on excess frequency of a second user-defined cutoff, is calculated and the mean score taken over all the product molecular weights.

The diversity of the library encoded in a single chromosome, when judged by the individual substituent sets, is assessed by considering the number of times each 3D family or 2D cluster occurs in each of the sections of the chromosome. These diversity calculations are conducted separately on each R position. However, it would also be possible to combine the precursor sets for all positions and cluster or partition on one common chemical space.

In the library design problems to which the 3D diversity measure has so far been applied, the number of families present in a set of precursors has always exceeded the number of substituents required in that position in the library design. In this case a simple diversity score can be produced for each section of the chromosome by imposing a penalty related to the number of repeated occurrences of any given family or cluster. At best the number of families that could be represented in a section of a chromosome is equal to the number of genes set in that section, so the penalty score is based on the ratio of the number of repeated families to the number of bits set. The mean penalty score is taken across all sections to produce a diversity score for the chromosome. Scoring has to be normalized by the number of set genes in each to avoid biasing the diversity score by the number of precursors selected. Without this normalization the algorithm will tend to reduce family repeats by reducing the number of precursors selected. Note that if subsequent design problems arise in which there are fewer groups than required substituents, it will be straightforward to devise a score based on the evenness of the frequency distribution of group numbers across the selected substituent set.

If the objective is to assess the diversity of the products, a frequency distribution of the cluster numbers or cell numbers is constructed for all the library products represented in the chromosome. Given a user-defined acceptable cluster or cell redundancy, a score is calculated for the excess frequency using the method described above for the molecular weight distributions.

As was the case for library II, it is sometimes required to maximize the number of products in a library, while keeping the deconvolution problem within acceptable bounds. In this case a penalty score is calculated by taking the mean across all sections of the chromosome of the square of the number of unset genes in each section.

Having produced a score for each factor in the optimization, a weighted mean of these component scores is taken to be returned as the fitness of the chromosome. The relative weights on each individual factor are user-defined and can be adjusted to place more emphasis on some aspect or aspects of a given design problem.

Step 4. Selection. A mating pool of size equal to the required number of replacements for a generation is created by roulette wheel selection, giving a greater chance to more fit members of the population to enter the mating pool. Rather than use the raw scores for this selection, our preliminary experiments showed it to be more effective to base the selection on rank.[29] The population is ranked according to the scores, and selection for entry into the mating pool is based on a function of the rank rather than the score itself. Experiment showed both linear and nonlinear rankings to be equally useful. In the former the normalized fitness is a linear function of the rank; in the latter the spacing is geometric.

Step 5. Reproduction. *Crossover* is the genetic algorithm operator which swaps genetic material between parents to produce children. Two parents are first selected at random from the mating pool to produce children. For each gene in child 1, *uniform crossover*[30] selects randomly whether that gene will be inherited from parent 1 or parent 2; child 2 inherits the corresponding genes from the other parent. *Two-point crossover* selects two gene positions in the parents. Child 1 inherits the genes between these positions from parent 2 and outside them from parent 1. Once again, child 2 is the complement of child 1. In preliminary experiments uniform crossover was generally found to be more successful than two-point crossover. A crossover rate of around 0.8 is used, meaning that having selected two parents there is an 80% chance that crossover will be applied and a 20% chance that the parents will pass into the child pool directly. *Mutation* by simple inversion is applied to the children produced. Genes are selected at random, typically at a rate of around 1 or 2 genes/chromosome, and the bit flipped from 1 to 0 or *vice versa*.

Step 7. Replacement. The genetic algorithm has been run using both *generational replacement* and *steady-state replacement*. In the former, one parent generation gives rise to a whole child generation, and the parent generation is then either conditionally or unconditionally replaced by its children. In the latter case, as soon as a single child is produced, it is then conditionally or unconditionally inserted into the generation, and so genetic material from the child is immediately available to influence the production of the next child, which it would not be in the former. Our initial experiments showed that while generational replacement occasionally gives better solutions, the steady-state method gives much faster convergence and answers which are never too much worse than generation replacement.

Experiment showed conditional replacement to be most effective; that is, child chromosomes only enter the population if they are an improvement on existing members of the population, meaning that weaker members are replaced at each stage. *Elitism* is used to unconditionally carry forward the best member of each generation into the next.

Library Design Results

Library I, discussed in the introduction and shown in Scheme 1, had two sites of diversity. The first substituent position, R_1, was produced from carboxylic acids, chloroformates, and amines and the second, R_2, from carboxylic acids, aldehydes, and carbamoyl chlorides. A search of the Available Chemicals Directory (ACD),[31] produced 360 precursors compatible with the chemistry for R_1 and 259 for R_2. There are therefore 93 240 possible library product compounds. The design of this library called for the production of 10 000 compounds by combining 100 R_1s with 100 R_2s. Deconvolution was to be by high-resolution experiment; therefore, the minimum redundancy was required for every unique molecular formula. Diversity was assessed on the precursor sets, and so precursors were transformed into substituents, and 3D families, based on common patterns of 3D pharmacophore points, were precomputed for each R position separately yielding 203 families for R_1 and 155 families for R_2.

The following results are for the genetic algorithm running in steady-state mode with a population of 100. The chromosome was defined to allow 100−150 genes for both R_1 and R_2. The frequency cutoff was set at 2; *i.e.*, a frequency of 3 contributes 1 to the total excess frequency. Equal weigh was placed on diversity and deconvolution in the fitness function. Random initialization was used for the chromosomes.

Figure 3 shows the variation of total excess frequency (TEF) and the total number of families present for the best chromosome over a typical run. Note that the TEFs quoted in this section are not squared although the squared value was used by the fitness function. Figure 4 shows the molecular formula frequency distributions for generations 1 and 1000 and at convergence, which for this run was after 392 000 generations. On average, genes with fewer bits have a lower TEF score, so these come to dominate the population after the first few thousand generations. Following rapid improvement to the molecular frequency distribution, the remainder of

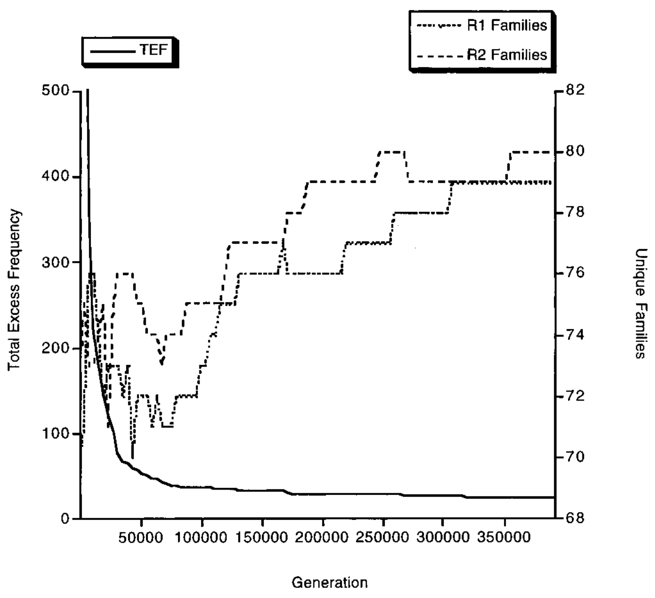


Figure 3. Evolution of library I. The total excess frequency (TEF) of molecular formulæ and the total number of 3D families present in the precursor sets for R_1 and R_2 at each generation are shown.

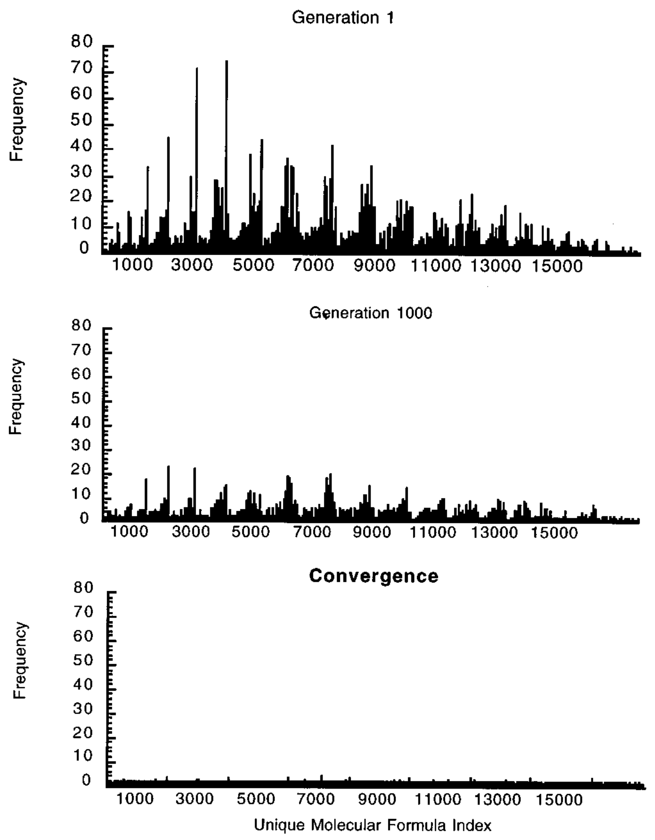


Figure 4. Evolution of library I. The molecular formula frequency distribution is given for the best chromosome in the first, one-thousandth, and last generations.

the run produces small improvements in the TEF while increasing the diversity of the library in terms of the number of different families represented by the 100 precursors at each diversity site. Over all 17 000 unique molecular formulae, only 23 exceed the cutoff after the program converges.

The typical run time for the program, using a Silicon Graphics Indigo2 running at 250 MHz, is approximately 0.05 s CPU/generation in steady-state mode, of which over 0.04 s CPU is spent in the fitness function. The total time to convergence for this library was approximately 5.5 h. Examination of Figure 3 suggests that reasonable approximate solutions can be obtained in a much shorter time since the most rapid improvement to the score comes early in the run. Furthermore, in cases where no good solution can be found, possibly because the frequency cutoffs have been set unrealistically low, this is often apparent early in the run when little improvement will be made during the first generations. In this case the program can be terminated and rerun with adjustments to the GA parameters and frequency cutoffs.

Table 1. Variation in the Components of the Fitness Function, Total Excess Frequency (TEF) of Molecular Formulae and Number of 3D Families, over 10 Runs for Library I, Resulting from the Stochastic Nature of the Genetic Algorithm[a]

		families	
run	TEF	R_1	R_2
1	23	78	84
2	25	81	77
3	20	81	80
4	25	79	77
5	22	80	78
6	26	74	78
7	23	79	79
8	24	81	79
9	25	81	80
10	23	81	80

[a] Only the seed to the random number generator is changed between runs.

Table 2. Variation in the Total Excess Frequency (TEF) of Molecular Weight and Number of 3D Families over Six Runs of Library I, Using Initialization of the Population for Maximum Diversity

		families	
run	TEF	R_1	R_2
1	32	87	86
2	28	84	83
3	27	90	82
4	33	83	86
5	36	85	86
6	28	88	83

Since the GA method is stochastic, rerunning the program with the same parameters will produce different results. As long as the program is not converging prematurely, it should be expected that the final solutions will have approximately equal fitness scores, however. Table 1 shows the TEF and number of families for the best chromosome at convergence for 10 runs of the program varying only the seed to the random number generator. The table shows that there is little variation in the TEF (between 20 and 26) or in the total number of families represented in R_1 (74–81) or R_2 (77–84).

The effects of initializing the chromosomes for maximum diversity rather than entirely at random are shown in Table 2. These runs have produced a similar outcome, although with more diversity at the expense of a slightly worse deconvolution profile. Typically in the first generation of these runs, the TEF scores were in the 3000–5000 range, and so the final TEFs of these runs and those with the random initialization are seen to be very similar. The number of generations to convergence was similar in both sets of runs. Finding similar outcomes with two different types of starting points is a further indication that the GA is not converging prematurely on solutions far from the global minimum.

Since the chromosomes are initialized for maximum diversity in these cases and only changed to satisfy the molecular formula considerations, it can been seen that approximately 10−20% of the possible diversity at each R position has had to be sacrificed to satisfy the deconvolution constraint.

For library II (see Scheme 2), three sites of diversity were available on a small molecule core. Positions R_1 and R_3 were derived from aldehydes and R_2 from amino acids. In the existing inventory of precursors available within Abbott at the time at which the library was constructed, there were 53 suitable candidates for the aldehydes and 42 for the amino acids. The library was not to be of a predetermined size, but instead, a maximum allowed redundancy of 100 was specified for any one molecular weight and the largest possible size library required. In addition, within any one product molecular weight a limit of 10 redundancies at any one substituent weight was imposed. The precursors had already been selected for diversity so there was no need to optimize the number of families present in this case. After experiment, weights were picked for the different parts of the scoring function to ensure that the TEF was consistently at or very close to 0 and that the sum of the monomer TEFs was always 0. These weights were 10:2:1 for monomer TEF:products TEF:library size. Upper and lower bounds of 10 and 50 monomers were applied to positions 1 and 3 and 10 and 42 to position 2.

Figure 5 shows the molecular weight distribution of the best chromosome at generation 1 and at convergence. Since some chromosomes were generated with very few bits, some libraries in generation 1 have an extremely low TEF; however, they contain very few compounds and so have a poor score due to this latter factor. Table 3 shows the variation in TEF and total number of compounds in the library over 10 runs varying only the seed to the random number generator. The results suggest the optimal library size to be between 21 672 and 24 300. In addition it can be seen that the best results are always obtained using a large number of precursors at R_1 and R_3 relative to the total available (43−45 of 53) and a much smaller number at R_2 (11 or 12 of 42). This is an indication that there is most molecular weight redundancy in the precursor set for R_2 and that the best way to produce a larger library while staying within the deconvolution limits would be to identify more candidate precursors for R_2.

A final example, library III, required evaluation of the diversity of library products rather than the substituent sets. Two diversity sites were available on an asymmetric core with the same set of 315 acids from which to choose at both positions. The chemistry shown in Scheme 1 was used to construct this library. The library size was to be 100 × 100. The 99 225 possible library products had 13 839 unique molecular formulae and were clustered, using Ward's agglomerative clustering and MACCS 2D fingerprints, into 10 000 clusters so that the most diverse set possible would have no more than one product from each cluster. In the fitness function equal weight was placed on the cluster and molecular formula redundancies. Initialization was by random assignment. Figure 6 shows the frequency of occurrence

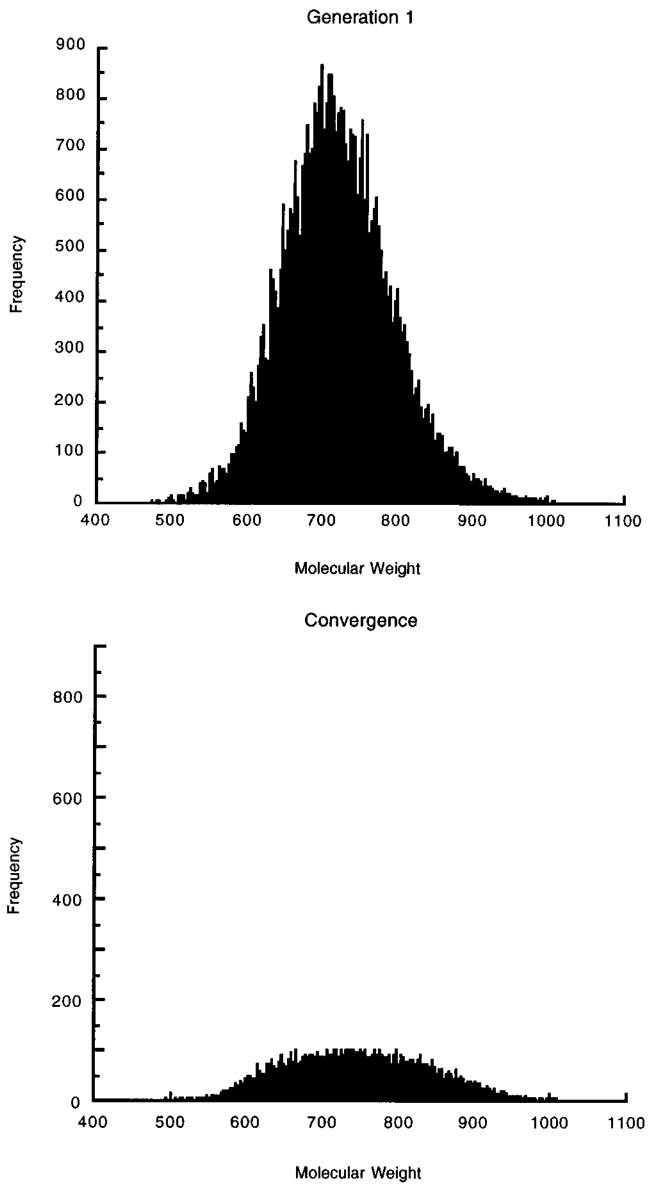


Figure 5. Integral molecular weight frequency distribution of the best chromosome in the first and last generations for library II.

Table 3. Variation in the Number of Precursors at Each Diversity Position and the Total Number of Compounds in the Library over 10 Runs for Library II[a]

run	R_1	R_2	R_3	total compounds
1	45	12	45	24 300
2	44	11	45	21 780
3	43	12	42	21 672
4	43	12	44	22 704
5	43	12	45	23 220
6	45	11	46	22 770
7	44	12	43	22 704
8	43	12	43	22 188
9	43	12	44	22 704
10	44	12	44	23 232

[a] Only the seed to the random number generator varies between the runs.

of each cluster number in the products for the best chromosome in generation 1 and at convergence, applying a redundancy cutoff of 4. Figure 7 shows the molecular formulae redundancies for the same chromosome, applying a cutoff of 3. Table 4 shows frequency counts for the total number of occurrences of each cluster size. Between generation 1 and convergence the

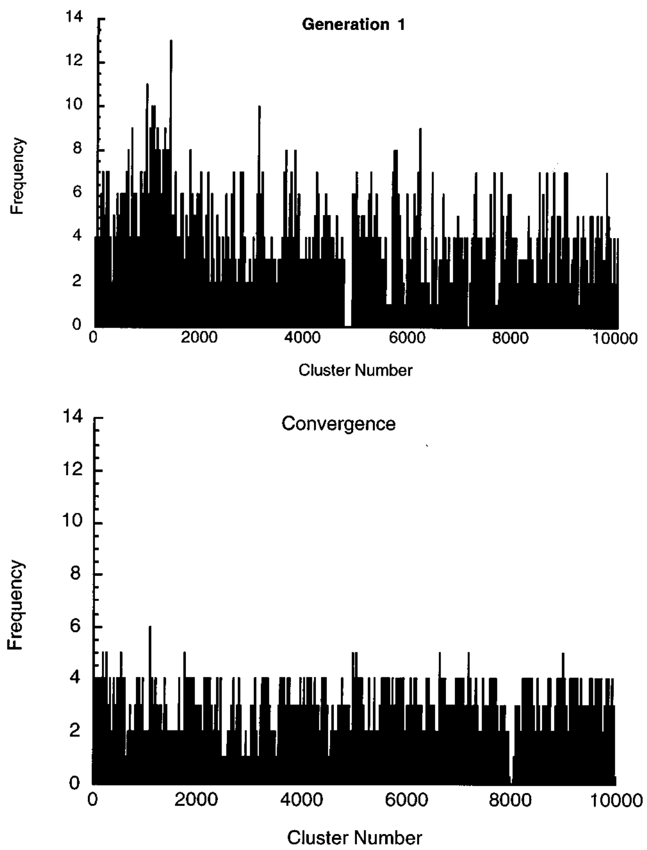


Figure 6. Cluster frequency distribution for the library products in the best chromosome of the first and last generations for library III.

Table 4. Evolution of the Diversity Score for Library III[a]

cluster size	generation 1	convergence
0	4794	3835
1	2017	3404
2	1689	1913
3	784	633
4	406	205
5	146	9
6	83	1
7	49	0
8	19	0
9	8	0
10	3	0
11	1	0
12	0	0
13	1	0

[a] The number of clusters of each cluster size is given for the solution encoded by the best chromosome in the first and last generations. Larger clusters indicate more redundancy among the library products. Under-representation of small clusters indicates a lack of coverage.

number of clusters represented many times is greatly reduced, while the number of clusters represented between zero and two times is increased, indicating an increase in diversity and also coverage. At convergence there are 3835 clusters not covered by compounds in the library and 5860 unrepresented molecular formulae. The total diversity that has had to be sacrificed both to the combinatorial constraint and to the deconvolution criteria is therefore seen to be just under 40% of the theoretical maximum (*i.e.*, 3800 of 10 000 clusters). The variation in the two TEF statistics over five runs is shown in Table 5. The first run in this table is the one discussed above. Once again there is little variation in

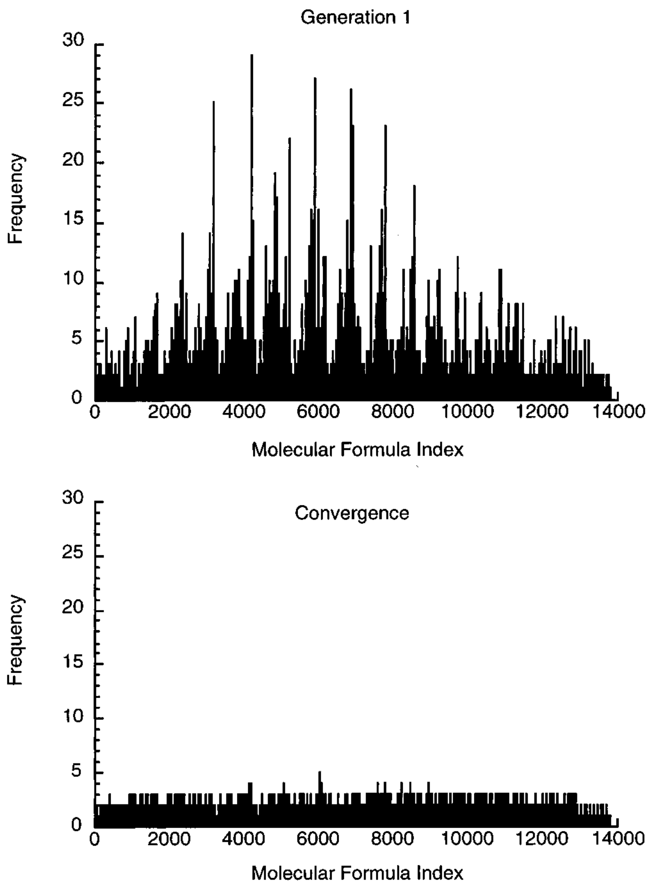


Figure 7. Molecular formula frequency distribution for the best chromosome in the first and last generations for library III.

Table 5. Variation in the Total Excess Frequency (TEF) of the Clusters and Molecular Formulae over Five Runs of Library III Changing Only the Seed to the Random Number Generator

	TEF	
run	cluster	molecular formula
1	10	13
2	20	15
3	14	13
4	11	12
5	17	13

the overall diversity or molecular formula redundancy among the solutions.

Since the product diversity is being assessed rather than that of the substituents, it is not possible to initialize the chromosomes for maximum diversity. However, an initialization biased toward diversity was investigated in which the substituents were clustered, using MACCS keys and Ward's clustering, and chromosomes initialized by selecting no more than one member of any cluster in any chromosome. Over five runs there was no difference between the final product cluster TEFs or molecular formula TEFs for these runs and for those which started with random initialization.

Discussion

The GALOPED method permits the design of combinatorial mixture libraries that simultaneously meet a number of design criteria. Efficient deconvolution of active compounds by mass spectroscopy is allowed for by designing libraries with the minimum redundancy of molecular weights or molecular formulae and optionally minimum redundancy among substituent molecular weights at any one product molecular weight. Diversity among the library products is either accounted for directly from product cluster numbers or based on a selection of precursors that maximizes the number of different pharmacophore point patterns present among each set of substituents. There are several advantages to the genetic algorithm method:

It is very fast and needs only to examine a tiny fraction of all potential libraries to find satisfactory solutions. In library I for example, under 400 000 of the possible 10^{164} solutions are considered. It is, of course, not possible to say how close the program may come to an optimal solution since this would require the enumeration of all possible libraries. However, it seems clear from examining the solutions present in the initial population of any run that it would not be possible to find good solutions simply by randomly generating possible libraries. Given the types of chemistry currently being developed for solid support and the resources of the Available Chemicals Directory, it is easy to imagine that much larger candidate sets of precursors than those given in the examples in this paper may need to be processed. For example, a selection from 1000 × 1000 × 1000 choices may not be unrealistic. This increase in size on its own should not impose a limitation on the method. Indeed for a fixed final library size, more candidates at each position might allow for more choices when attempting to maximize the diversity. Other factors will be more important to the speed of convergence of the algorithm. The size of the final design library will affect the speed of the fitness function since all possible substituent combinations have to be enumerated and evaluated. A second factor will be the amount of variability available in the products. If unrealistic cutoffs for cluster and molecular weight redundancy are set in relation to the molecular weight and cluster profiles of all potential library products, then convergence on good solutions will be much more computationally difficult to achieve.

The method is able to optimize a number of often competing factors simultaneously, and it is readily extensible to include new design factors. The fitness function is independent of the searching algorithm and is the only part of the program which has knowledge of the problem domain. This means that it is straightforward to modify the former without affecting the latter. We have demonstrated optimizing the diversity of either the precursor sets or the product structures. However, any other properties of either could equally well be considered instead of, or in addition to, diversity. One can imagine many properties which might be optimized, for example, physical properties such as log *P* or the calculated affinity of the library products for a particular receptor. In addition it is equally simple to write functions to maximize the spread of properties or confine them to a required range. The only restriction to the use of library product properties is that there must be sufficiently few that they can be enumerated and the properties calculated. While one could imagine calculating product properties on-the-fly for only those structures which are included in any chromosome, analysis of many GALOPED runs suggests that almost all possible library products will be examined at least once. There would therefore be little to be gained from on-the-fly enumeration for our current examples. For much larger libraries it may be that a lower percentage of all possible library products would appear in any chromosome, in which case on-the-fly enumeration may become advantageous. However, the speed of enumeration of the required properties would be a limiting factor on their use with very large virtual libraries, and we may have to be content with simultaneously optimizing properties of the precursor sets in these cases.

Table 6. Number of Precursors in Common between the Top Five Different Solutions from a Single Run for Library III[a]

	1	2	3	4	5
1		100	99	99	100
2	99		99	99	100
3	100	99		98	99
4	100	99	100		99
5	99	99	99	99	

[a] Values above the diagonal refer to precursors for diversity site R_1, below the diagonal to precursors for R_2.

Table 7. Number of Precursors in Common between the Best Solution from Five Separate Runs for Library III[a]

	1	2	3	4	5
1		48	58	41	44
2	54		52	52	48
3	57	51		42	53
4	57	48	47		49
5	46	48	46	55	

[a] Values above the diagonal refer to precursors for diversity site R_1, below the diagonal to precursors for R_2.

The method can provide many different solutions to a problem from which the combinatorial chemist can choose. These might be either different chromosomes from the same population or the best chromosome from many populations obtained running the program several times. Table 6 shows, for the best five different chromosomes in a single run of library III, the number of precursors in common between each pair of solutions. Above the diagonal are the values for position R_1, below the diagonal for R_2. The suggested libraries are different by only one or two precursors, showing that the solutions in a single run represent slight variations on a common theme. Table 7 shows the equivalent results for the best chromosome over five separate runs of library III. In contrast to the results in Table 6, this has resulted in the production of very different solutions, albeit with approximately equal fitness. In either position any pair of solutions only has approximately one-half of the precursor sets in common. Structures of the precursor sets selected in these five runs are provided as Supporting Information.

Conclusions

The method we have described in this article allows us to design combinatorial mixture libraries optimized for biological screening for lead identification. It also provides the framework for the design of libraries to meet any calculable optimization function.

Supporting Information Available: Chemical structures of the precursor sets selected in five runs referred to in Table 7 (70 pages). Ordering information is given on any current masthead page.

References

(1) Brown, R. D.; Bures, M. G.; Martin, Y. C. Similarity and Cluster Analysis Applied to Molecular Diversity Presented at the *209th American Chemical Society Meeting*, Anaheim, CA, 1995.

(2) Dunbar, J. Analyzing Molecular Diversity by Database Clustering. *Abstracts of Papers of the American Chemical Society*, American Chemical Society: Washington, DC, 1994; Vol. 208, 124-COMP.

(3) Shemetulskis, N. E.; Dunbar, J. B.; Dunbar, B. W.; Moreland, D. W.; Humblet, C. Enhancing the Diversity of a Corporate Database Using Chemical Database Clustering and Analysis *J. Comput.-Aid. Mol. Des.* **1995**, *9*, 407−416.

(4) Martin, E. J.; Blaney, J. M.; Siani, M. S.; Spellmeyer, D. C.; Wong, A. K.; Moos, W. H. Measuring Diversity: Experimental Design of Combinatorial Libraries for Drug Discovery *J. Med. Chem.* **1995**, *38*, 1431−1436.

(5) Martin, Y. C.; Brown, R. D.; Bures, M. G. Quantifying Diversity. In *Combinatorial Chemistry and Molecular Diversity*; Kerwin, J. F., Gordon, E. M., Eds.; Wiley: New York, in press.

(6) Turner, D. B.; Tyrrell, S. M.; Willett, P. Rapid Quantification of Molecular Diversity For Selective Database Acquisition. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 18−22.

(7) Ferguson, A. M.; Patterson, D. E.; Garr, C. D.; Underiner, T. L. Designing Chemical Libraries for Lead Discovery. *J. Biomol. Screening* **1996**, *1*, 65−73.

(8) Brown, R. D.; Bures, M. G.; Martin, Y. C.; Pavlik, P. A.3D Property Based Precursor Selection for Combinatorial Library Construction. Presented at the American Chemical Society, 210th National Meeting, Chicago, IL, 1995.

(9) Gillet, V. J.; Willett, P.; Bradshaw, J. The Effectiveness of Monomer Pools for Generating Structurally-Diverse Combinatorial Libraries. Molecular Graphics and Modelling Society, 1996.

(10) Brown, R. D.; Martin, Y. C. Use of Structure-Activity Data to Compare Structure-Based Clustering Methods and Descriptors for Use in Compound Selection. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 572−584.

(11) DiverseSolutions v2.0.1; Laboratory for Molecular Graphics and Theoretical Modelling, College of Pharmacy, University of Texas at Austin, TX.

(12) Agrafiotis, D. K. Stochastic Algorithms for Maximizing Molecular Diversity. ECCC3 - The Third Electronic Computational Chemistry Conference, 1996.

(13) Cummins, D. J.; Andrews, C. W.; Bentley, J. A.; Cory, M. Molecular Diversity in Chemical Databases: Comparison of Medicinal Chemistry and Knowledge Bases and Databases of Commercially Available Compounds. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 750−763.

(14) Special Report - Combinatorial Chemistry. *C&EN* **1996**, February 12, 28−73.

(15) Loo, J. A.; DeJohn, D. E. Application of Mass Spectrometry for Characterizing and Identifying Ligands from Combinatorial Libraries. In *Annual Reports in Medicinal Chemistry*; Bristol, J. A., Ed.; Academic Press: San Diego, CA, 1996; Vol. 31.

(16) Choong, I. C.; Ellman, J. A. Solid-Phase Synthesis: Applications to Combinatorial Libraries. In *Annual Reports in Medicinal Chemistry*; Bristol, J. A., Ed.; Academic Press: San Diego, CA, 1996; Vol. 31.

(17) Mitchell, M. *An Introduction to Genetic Algorithms*; The MIT Press: Cambridge, MA, 1996.

(18) Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: Reading, MA, 1989.

(19) Holland, J. H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, 1975.

(20) SUGAL (Sunderland University Genetic ALgorithm package): Ver 2.1; Dr. Andrew Hunter, Sunderland University; http://www.trajan-software.demon.co.uk/sugal.html.

(21) Singh, J.; Ator, M. A.; Jaeger, E. P.; Allen, M. P.; Whipple, D. A.; Soloweij, J. E.; Chowdhary, S.; Treasurywala, A. M. Application of genetic algorithms to combinatorial synthesis - a computational approach to lead identification and lead optimization. *J. Am. Chem. Soc.* **1996**, *118*, 1669−1676.

(22) Sheridan, R. P.; Kearsley, S. K. Using a genetic algorithm to suggest combinatorial libraries. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 310−320.

(23) Weber, L.; Wallbaum, S.; Broger, C.; Gubernator, K. A Genetic Algorithm Optimising Biological Activity of Combinatorial Compound Libraries. *Angew. Chem.* **1995**, *107*, 2453−2454.

(24) Project Library; MDL Information Systems, Inc., San Leandro, CA.

(25) Chem-X; Chemical Design Ltd., Chipping Norton, Oxfordshire, U.K.

(26) Tmftalk. Part of the Daylight contributed code directory; Dr. B. Rohde, Ciba-Geigy, Basel, Switzerland.

(27) Patterson, D. E.; Cramer, R. D.; Furguson, A. M.; Clark, R. D.; Weinberger, L. E. Neighborhood Behavior: A Useful Concept for Validation of Molecular Diversity. *J. Med. Chem.* **1996**, *39*, 3049−3059.

(28) Brown, R. D.; Bures, M. G.; Danaher, E.; Lico, I.; Martin, Y. C.; Pavlik, P. A.; Cesarone, J.; Chen, R.; Delazzer, J.; Hawe, W.; Liu, M. Structure Based Diversity Selection of Precursors for Combinatorial Libraries. In Molecular Graphics and Modelling Society - EC1, 1996.

(29) Whitely, L. D. The Genitor Algorithm and Selection Pressure: Why rand-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*; Schaffer, J. D., Ed.; Morgan Kaufmann: San Francisco, CA, 1989.

(30) Syswerda, G. Uniform Crossover in Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*; Schaffer, J. D., Ed.; Morgan Kaufmann: San Francisco, CA, 1989.

(31) Available Chemicals Directory; MDL Information Systems, Inc., San Leandro, CA.

(32) Meyer, J. P.; Zhang, J.; Bjergarde, K.; Lenz, D. M.; Gaudino, J. J. Solid Phase Synthesis of 1,4-Benzodiazepine-2,5-diones. *Tetrahedron. Lett.* **1996**, *37*, 8081−8084.

JM970033Y