Combinatorial Library Design: Maximizing Model-Fitting Compounds within Matrix Synthesis Constraints

Robert V. Stanton, John Mount, and Jennifer L. Miller

Molecular Design Group, CombiChem, Inc., 1804 Embarcadero Road, Suite 201, Palo Alto, California 94303

Received September 2, 1999

The use of combinatorial chemistry has become commonplace within the pharmaceutical industry. Less widespread but gaining in popularity is the derivation of activity models from the high-throughput assays of these libraries. Such models are then used as filters during the design of refined daughter libraries. The design of these second generation libraries, which efficiently test and conform to the derived activity model from the large space of virtual possibilities, remains an area of considerable research. We present here a computationally efficient method for the design of optimally dense (in model matching compounds) synthetic matrices from *in silico* virtual libraries.

INTRODUCTION

Combinatorial chemistry[1−3] is increasingly being used by the pharmaceutical industry as a powerful tool to speed the process of drug discovery and optimization.[4−7] This technology has also profoundly reshaped the opportunities for computational input during drug discovery and lead optimization efforts. The application of computational methods has quickly expanded from suggestions for tens of compounds, as might come from structure based design, to thousands and even tens of thousands of compounds, in the form of combinatorial library designs. The need to propose thousands of compounds for synthesis and screening has demanded the development or extension of several computational design techniques such as diversity,[4,6] informative design,[8] clustering,[9] docking,[10−12] and 3-D searching.[4,6,13]

The use of an activity model as a computational filter is an example of a computational methodology which is still evolving to meet the needs of combinatorial chemistry. This technique was originally applied to small sets of compounds, and it has been used successfully to identify a number of novel inhibitor classes.[14−18] Such models attempt to represent what is known, suspected, or might be inferred about what is necessary for activity against a specific target. An activity model may be created in many different ways using pharmacophores, excluded volumes, key features, or even crystallographic data. Indeed, there are many commercial and proprietary software packages devoted to just this task.[19−21] However, two obstacles remain with the extension of this technique to the design of synthetic combinatorial libraries.

The first obstacle comes from the application of these filters to extremely large numbers of compounds. Activity models, which were developed to operate on small numbers of compounds, must now deal with the explosion in product space that is inherent in any combinatorial chemistry approach. For example, a combinatorial library created by peptide bond formation has a potential size of more than 25 million products if only the >5000 acids and >5000 amines from the Available Chemicals Directory (ACD) are considered. From this library of ∼25 million only a relatively small number of compounds can be synthesized, usually in the range of ∼1000 or less. It is this choice of which 1000 compounds to make (from the 25 million source pool) that computational methods address. The development of less complex computational filters has been necessary to address this problem. A number of both academic and commercial programs have been developed recently which attempt to solve this problem using various approaches.[22−24]

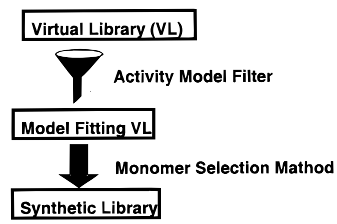

Figure 1. Schematic representation of the steps involved in creating a synthetic matrix library from a virtual library.

The second obstacle comes from the experimental constraints of combinatorial synthesis. Instead of selecting one, two, or a handful of compounds for synthesis, computational methods now have to select a larger set of compounds for *matrix* synthesis. Synthetic combinatorial libraries are generally designed with limits on both the total number of compounds synthesized (usually a set of 96-well plates) and on the number of monomers used. All products in a row or column of the plate contain the same monomer; this is the essence of the term “matrix constraint”. This constraint typically operates downstream of an activity model (Figure 1). That is, even when an activity filter is applied to a virtual combinatorial library, unless the filter is very stringent, one is still left with many more compounds than can be synthesized. It is a very challenging computational problem to determine which subset of these compounds is both optimally dense in “model-matching” compounds and amenable to the experimental matrix constraints.

One note of caution is that the use of matrix synthesis and activity models may greatly sacrifice the diversity of the resultant synthetic library. If “one-off” compounds were made instead of a matrix synthesis, the library could be

10.1021/ci990183c CCC: $19.00
Published on Web 03/30/2000

	C_1	C_2	C_3	...	C_y
R_1	P_{11}	P_{12}	P_{13}	...	P_{1y}
R_2	P_{21}	P_{22}	P_{23}	...	P_{2y}
R_3	P_{31}	P_{32}	P_{33}	...	P_{3y}
⋮	⋮	⋮	⋮		⋮
R_x	P_{x1}	P_{x2}	P_{x3}	...	P_{xy}

Figure 2. Products, P_{xy}, formed from monomers R_x and C_y, with either model matching (1) or not (0). A successful detection algorithm will find the maximally dense submatrix of 1's.

"cherry picked" from the identified compounds (model fitting or otherwise). Additionally, a diversity metric could be used to ensure selection of a maximally diverse set. The benefits of such a library need to be weighed against the possible difficulties of one-off synthesis and the greatly increased number of reagents necessary. Computational schemes that are not dependent upon explicitly testing each member of a library, such as genetic algorithms[25] or simulated annealing,[7] have been developed and used with good results for synthesis not held to matrix constraints.

Figure 1 depicts the two separate challenges outlined here. The first is filtering/testing the virtual library against the activity model. The second is the subject of this paper and is the matrix design of a synthetic library which is maximally dense in model-matching compounds. We present here results from two solutions to the problem of designing a small combinatorial matrix from a much larger source pool of products of a combinatorial reaction. The first, a branch and bound technique, is the more rigorous but becomes computationally intractable even with moderately sized libraries. To overcome this problem, we also present a "cut-down" approximation to the branch and bound solution and show that the results, with respect to a real combinatorial library and activity model, closely approximate that of the true solution.

METHODOLOGY

Matrix Constraints. A synthetic combinatorial matrix is composed of products (P_{xy}) which have been formed from monomers (R_x and C_y). An example of a generic combinatorial matrix is shown in Figure 2. The products of such a matrix can be tested against a derived activity model, giving each individual compound a binary matching or not-matching score. Looking at a combinatorial matrix such as that in Figure 2 reveals the necessity of designing in monomer space (R_1 to R_x, and C_1 to C_y) to obtain a product space (P_{11} to P_{xy}) of interest. That is to say, only by careful selection of monomers can the competing design criteria of library size and percentage of model-matching products be balanced.

Designing small synthetic matrix libraries from the space of all possible products for a combinatorial reaction, when reduced to its simplest form, can be thought of as a problem of clique detection.[26] The combinatorial matrix, once tested against the model of interest, becomes a sparse binary matrix (i.e. composed of only 1's and 0's) with the 1's corresponding to the model-fitting compounds and the 0's compounds that do not fit the model. Finding a maximally dense (i.e. containing the most 1's) submatrix in such a system can be done using a branch and bound algorithm.[27] We have implemented this relatively expensive solution and use it for comparison to our more approximate but computationally more expedient cut-down method, described later.

Branch and Bound. Given a sparse matrix as a starting point, the branch and bound algorithm selects which rows and columns to include in a final, maximally dense submatrix. The choice of including any particular row or column can be thought of as a "branch", and the exclusion as a "bound". In the limit of no bounds, all possible submatrices are considered (guaranteeing both a good result and horrible run time). During the selection process this algorithm often reconsiders its decisions (branches) by considering a bound heuristic. The bound heuristic is a routine that looks at a current selection of rows and columns and returns an upper bound on how dense a complete solution of which they can possibly be a part. Thus, as is often the case, when the branch and bound algorithm has found a good tentative solution and a new partial solution being investigated turns out (by way of the bound) to have no chance of being any better, the search is curtailed. As long as the bound returned is correct, the algorithm still finds an optimal solution. If the bound is not always correct (and is merely heuristic in nature), then the branch and bound algorithm may return a solution that is not the best possible.

A trivially correct bound is to always assume that any partial selection of rows and columns can be extended into a matrix of density 1. An example of this would be to assume that a given 3×3 matrix which contained 2 desirable products could be extended into a 5×5 matrix which contained 25 desirable products. This is patently false. This bound, while possible, is not particularly useful as it never allows the branch and bound algorithm to skip any of its search. A slightly more realistic bound is given by assuming we can add perfect rows and columns to any matrix. An example would be to assume any 3×3 matrix containing 2 desirable products can be extended into a 5×5 matrix containing $5 \times 5 - 3 \times 3 + 2 = 18$ desired products. This would also be correct, yet very inefficient. So, in addition to being correct we wish for our bounds to be tight. That is, we would like correct bounds that tend to be small. One way to do this would be to use a search to find which complete solutions a partial selection of rows and columns can be expanded into. However, this involves just as much work as the branch and bound algorithm itself. When confronting such an intractable problem, the traditional approach is, instead of solving the problem at hand, to solve a different problem that has been chosen to be easier and quicker to solve. This technique is called a relaxation. Below we describe the relaxation we employed in our cut-down method.

Relaxation of the Problem. Our original problem is to select an actual small matrix (for synthesis) from a larger matrix (of potential products). Any such matrix can be described as the intersection of a set of rows and columns from the original larger matrix. For each row we reserve a variable R (i.e. R_1, R_2, ..., R_x). We restrict the R_i to be zero or one. When row I (which, remember, denotes a monomer selection) has been selected for synthesis, we record this by setting R_i to 1; if row I has not been selected, we record this by setting R_i to 0. Similarly we reserve a set of variables C_1, C_2, ..., C_y to record our column selections. We further assume, by simulating the possible reactions and scoring the product molecules, that we have ready constant $P_{i,j}$ such that $P_{i,j}$ has been set to 1 if we (in silico) like the molecule found in the ith row and jth column and 0 otherwise. It is important to

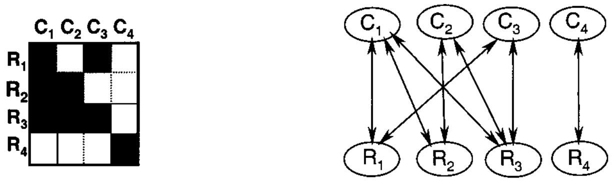

Figure 3. Binary matrix and its graph theory representation. Each node represents a model matching compound (colored black) and each edge the monomer relationship between that product and other model-matching products.

note the following: the R's and C's are variables and the P's are constant (determined by our software).

In this notation we see that a library using m row monomers and n column monomers (yielding $m \times n$ products) is exactly a setting of the R and C variables such that

$$\sum_{i=1}^{X} R_i = m \quad \text{and} \quad \sum_{i=1}^{Y} C_i = n \tag{1}$$

(Remember R_i and C_j are restricted to each be zero or one.) The number of desired molecules picked up in a design given by a particular setting of the R and C variables is then just

$$\sum_{i=1}^{X}\sum_{j=1}^{Y} P_{ij} R_i C_j \tag{2}$$

Only molecules that are in both a selected row and column ($R_i \neq 0$ and $C_j \neq 0$) and "desired" ($P_{ij} \neq 0$) contribute to the sum.

This leaves us with the abstract problem of, for a given set of constants P, finding settings for the variables R and C that maximize the sum (2). To "relax" the problem, we drop one or more of the constraints. This produces a bound because any solution of the original problem is also a solution to the relaxed one; the optimal value can only go up. The trick is to drop enough constraints so that the problem becomes easy, but not so many that the bound becomes meaninglessly large.

Our relaxed problem was to simply replace the condition that all R_i and C_j be each 0 or 1 with that they be real numbers in the range 0−1 (i.e. we now allow fractional values such as 0.5). While solutions to this mathematical problem can no longer be interpreted as selections of rows and columns, we can use them as bounds in our branch and bound search. This type of problem is called a quadratic program and in this case (positive definite) is easy to solve.[27]

Graph Theory. An alternative in attempting to find a computationally less intensive solution to the synthetic combinatorial library design is to approach it in terms of graph theory. By graph we mean a collection of items (nodes) that are joined in pairs (edges). For a matrix, we can consider each row and each column of our possible synthesis matrix as a node. If the compound given by the pairing of a row and column is considered good, we include an edge in the graph; otherwise we do not (Figure 3). In graph terms we wish to find a small set of nodes that has a larger number of edges within it. A graph with the largest possible number of edges (one for each pair of nodes) is called a "clique". The problem of finding a clique in a larger graph is called "clique detection" and is a well-known but difficult problem.[28] While our current problem is not to find a clique, it can be shown that it differs only in unimportant details. That is, if one had an efficient method for finding dense submatrices, we could use this method to find cliques in graphs. Because of this bidirectional relationship, and some significant notational similarities, we consider our problem very closely related to clique detection, which is the literature we looked to in search of solutions.[26,27]

Cut-Down Algorithm. Using the graph theory approach as inspiration, we approximate the exact solution by using a "cut-down" procedure instead of the selection, or "build-up", approach employed in the branch and bound technique. The cut-down method is what is called a greedy method[27] in that it eliminates the row or column that seems most advantageous to remove and never revisits its decision. Unfortunately, as both clique detection and near-dense matrix detection are both NP-hard, there is no guarantee that such a simple method will find the optimal solution. However, it is guaranteed that the matrix returned by this method will be amenable to synthesis (having the specified number of rows and columns) and will be at least as dense as the input data. In fact, the output matrix will always be among the most dense of all the matrices it considers on the way to a solution.

For simplicity, the cut-down method will be described here as a two-dimensional problem. The algorithm can be outlined as follows:

(1) Define the desired synthetic matrix dimensions.

(2) Examine the matrix and determine a score for each row and column (i.e. monomer) in the matrix. The score in its simplest form can be defined as the number of model fitting compounds present in the row or column.

(3) Beginning with the list of monomers (i.e. rows or columns) furthest from the desired length, remove the row (or column) which has the lowest score.

(4) Recalculate the score.

(5) Go back to step 2 until the matrix is of the desired size.

This method is intuitively attractive since it corresponds most closely to what would be done if the problem were presented manually. Benefits of this algorithm include that the resultant matrix will necessarily have an overall density of model-matching compounds greater than or equal to that of the initial matrix. Additionally, the method can accommodate rectangular designs and is easily extensible to higher order matrices and more complex scoring functions. Finally, this approach is intuitive and scales with the number of monomers. It is important to note that the recently developed PLUMS algorithm[29] described at the Fifth International Conference on Chemical Structures has several interesting similarities to the methodology described here, but remains distinct. PLUMS is an interesting example of "convergent evolution", as methodological problems are identified throughout the industry similar solutions are developed.[31]

A simple example of the cut-down approach is shown in Figure 4. The initial matrix **A** is shown with the scores for the individual rows and columns listed. If the desired matrix size is 3 × 3 and the initial matrix is 10 × 10, either axis may be examined first as neither is further from the final size than the other. After two rounds, one of row elimination and one of column elimination, matrix **B** is obtained. This process is continued, resulting in matrices **C**, **D**, **E**, etc., until

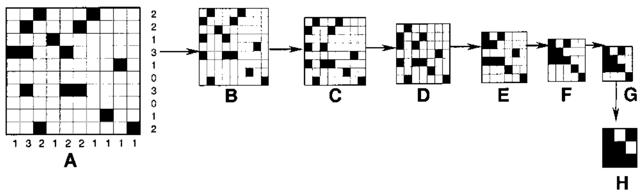

Figure 4. Example of the cut-down algorithm.

the final matrix **J** is determined. The density of model-fitting compounds in the original combinatorial matrix was 5%, while the final matrix contains a density of 90%. Attempting to optimize by hand the density of model-fitting compounds from even an extremely small 10 × 10 matrix is an interesting academic exercise (left to the reader) which quickly shows the difficulty to be expected as the size and dimensionality of the problem increase.

RESULTS/DISCUSSION

The results of applying the cut-down algorithm to both random matrices and real combinatorial libraries are summarized in Table 1. The first 12 examples are for a randomly generated binary matrix. This presents a pessimal case for clique detection; the random matrix assumes no relationship between row and column components and the product result. This is not the case in a combinatorial matrix containing model-matching compounds, as particular monomers (rows or columns) tend to be very correlated with "value" (model-matching). That is to say, if P_{11} matches an activity model, then it is likely (through the similarity hypothesis[25]) that other compounds formed with R_1 will also be model-matching. The next 3 examples in Table 1 (13−15) are for a generic combinatorial library (such as amines and acids in an amide bond formation) screened against activity models on an active project. The last 3 combinatorial examples (16−18) are from a three component dihydroisoquinilinone library.[30]

In example 1, given the random matrix of initial size 10 × 10 and initial density of 29%, a final "synthetic" matrix size of 4 × 4 was sought. The cut-down algorithm was able to find a matrix which was 50% dense in model fitting compounds (8 out of 16). The branch and bound algorithm with polynomial refinement was able to do slightly better, finding a 62% dense matrix (10 out of 16). For this small test, both methods performed nearly equivalently and neither was computationally prohibitive. The cut-down method took ~0.5 s on a large memory Pentium 400 MHz machine, while the branch and bound algorithm took ~1 s on the same machine.

The remaining examples (2−12) on random matrices compare the behavior of the branch and bound algorithm with that of the cut-down algorithm in cases where the computational expense of the branch and bound algorithm becomes more significant. We present these results by employing matrices of different initial sizes, densities, and desired final sizes. It is apparent from examples 1−12 that for a random matrix the cut-down method is able to find a dense submatrix within 5−10% of the density found by the more computationally expensive branch and bound technique. The differences in computational speed between the algorithms become more significant with either increased initial library size or dimensionality of the problem. In example 3, the branch and bound algorithm took ~7 min, 3 s to calculate the result, in contrast to the cut-down method which needed only ~0.5 s for the same system on an identical 400 MHz Pentium. A more stark example is the 500 × 500 matrix (example 10) in which the branch and bound algorithm took several hours to determine its result, while the cut-down algorithm only required ~1.4 s. Examples 11 and 12 contrast the methodologies performance for larger systems. In the case of the 5000 × 5000 array we were unable to calculate the branch and bound result within a reasonable time (<1 week). The 1000 × 1000 array took several days of CPU time (Pentium 400 MHz) for the branch and bound algorithm as compared to the ~2.1 s for the cut-down technique.

Most combinatorial libraries are (fortunately!) not random in their distribution of model fitting compounds. This creates product matrices which are much more amenable to the cut-down algorithm than random matrices. In the first "real-life" example (13), a library of 50 amines and 50 acids was fully enumerated and compared to a model derived for activity

Table 1. Data Comparing the Branch and Bound and Cut-Down Algorithms

example no.	matrix type	init size[a]	init dens[b]	final size[a]	cut-down final dens[c]	branch and bound final dens[c]
1	random	10 × 10	0.29	4 × 4	0.56	0.62
2	random	100 × 100	0.40	10 × 10	0.73	0.86
3	random	100 × 100	0.20	20 × 20	0.32	0.40
4	random	100 × 100	0.40	20 × 20	0.59	0.65
5	random	100 × 100	0.20	10 × 10	0.40	0.58
6	random	200 × 200	0.10	20 × 20	0.32	0.32
7	random	200 × 200	0.20	10 × 10	0.46	0.64
8	random	200 × 200	0.40	10 × 10	0.75	0.90
9	random	200 × 200	0.20	20 × 20	0.37	0.46
10	random	500 × 500	0.20	20 × 20	0.47	0.54
11	random	1000 × 1000	0.20	20 × 20	0.48	0.55
12	random	5000 × 5000	0.20	20 × 20	0.64	−
13	combinatorial	50 × 50	0.42	14 × 14	1.0	1.0
14	combinatorial	100 × 100	0.18	14 × 14	1.0	1.0
15	combinatorial	150 × 150	0.11	14 × 14	1.0	1.0
16	combinatorial	48 × 104	0.020	10 × 10	1.0	1.0
17	combinatorial	104 × 104	0.020	10 × 10	0.98	1.0
18	combinatorial	104 × 48	0.020	10 × 10	0.88	0.88

[a] The initial and final sizes of the matrices given in dimensions of the x and y monomer axis. [b] Initial density as defined by the number of model matching compounds divided by the total number of compounds in the library. [c] Final density found by the algorithm defined as the number of model matching compounds in the matrix defined by the final size.

(Table 1). One plate was proposed for the synthetic library (96-well format), and a 14 × 14 library was designed (the design was created slightly larger than the desired final library size anticipating the unavailability of some monomers). Our results show that both the cut-down and branch and bound methods were able to find complete submatrices (i.e. of 100% density). Examples 14 and 15 are for slightly larger initial library sizes using the same combinatorial library. The final synthetic matrix size calculated was the same as that in example 13, and the results again show nearly equivalent behavior. The final three examples (16−18) are taken from a three component library in which one element is held fixed in succession. The densities obtained from both algorithms are nearly identical for this system as well.

In conclusion, we have presented a method for the calculation of a highly dense synthetic submatrix of model matching compounds, given a sparse matrix of model-fitting compounds. While the cut-down method is shown to function adequately on random matrices as compared to a more precise branch and bound methodology, its strength is that it takes advantage of the correlation between similarity and activity found in real-life combinatorial library design efforts. In such cases, the cut-down method performs almost identically to the more computationally expensive branch and bound algorithm. The area in which the cut-down method is potentially most successful is its extension to large and higher order matrices, at which point the computational cost of other methods of clique detection become overwhelming. Additional benefits of the algorithm include that it is fairly intuitive and that additional factors such as price or similarity might be easily incorporated into the score used for each row and column.

ACKNOWLEDGMENT

The authors would like to thank David Miller for participating in initial discussions on this topic and Erin Bradley for one of the first applications of this work. Additionally, thanks go out to Peter Grootenhuis and David Spellmeyer.

REFERENCES AND NOTES

(1) Gallop, M. A.; Barrett, 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.; Barrett, 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.

(3) Terrett, N. K.; Gardner, M.; Gordon, D. W.; Kobylecki, R. J.; Steele, J. Combinatorial synthesis: The design of compound libraries and their application to drug discovery. *Tetrahedron* **1995**, *51,* 8135.

(4) Spellmeyer, D. C.; Blaney, J. M.; Martin, E. M. In *Computational Approaches to Chemical Libraries*; Charifson, P. S., Ed.; Dekker: New York, 1997; pp 165−194.

(5) Walters, W. P.; Stahl, M. T.; Murcko, M. A. Virtual screening: An overview. *Drug Discovery Today* **1998**, *3,* 160.

(6) Blaney, J. M.; Martin, E. J. Computational approaches for combinatorial library design and molecular diversity analysis. *Curr. Opin. Chem. Biol.* **1997**, *1,* 54.

(7) Spellmeyer, D. C.; Grootenhuis, P. D. J. Recent Developments in Molecular Diversity: Computational Approaches to Combinatorial Chemistry *An. R. Med. Chem. Rev.*, in press.

(8) Teig, S. *Informative Libraries are More Useful than Diverse Ones*. *J. Bio. Scr.* **1998**, *3*, 85.

(9) Willett, P. *Similarity and Clustering in Chemical Information Systems*; Research Studies Press: Letchworth, U.K., 1987.

(10) Kuntz, I. D.; Blaney, J. M.; Oatley, S. J.; Landridge, R.; Ferrin, T. E. *J. Mol. Biol.* **1982**, *161,* 269.

(11) DesJarlais, R. L.; Sheridan, R. P.; Dixon, J. S.; Kuntz, I. D.; Venkataraghavan, R. Docking flexible ligands to macromolecular receptors by molecular shape. *J. Med. Chem.* **1986**, *29,* 2149.

(12) Blaney, J. M.; Dixon, J. S. A good ligand is hard to find: Automated docking methods. *Perspect. Drug Discovery Des.* **1993**, *2*, 301.

(13) Brown, R. D. Descriptors for diversity analysis. *Perspect. Drug Discovery Des.* **1997**, *7*−*8,* 31.

(14) Bures, M. G. In *Recent Techniques and Applications in Pharmacophore Mapping*; Charifson, P. S., Ed.; Dekker: New York, 1997; pp 39−72.

(15) Pickett, S. D.; Mason, J. S.; McLay, I. M. Diversity profiling and design using 3D pharmacophores: Pharmacophore-derived queries (PDQ). *J. Chem. Inf. Comp. Sci.* **1996**, *36,* 1214.

(16) Van Drie, J. H. An inequality for 3D database searching and its use in evaluating the treatment of conformational flexibility. *Comput.-Aided Mol. Des.* **1996**, *10,* 623.

(17) Davies, K.; Briant, C. Combinatorial chemistry library design using pharmacophore diversity. *Net. Sci.* **1996**, *1* (http://llwww.netsci.org/Science/Combichem/feature05.html).

(18) Van Drie, J. H. Strategies for the determination of pharmacophoric 3D database queries. *J. Comput.-Aided Mol. Des.* **1997**, *11,* 39.

(19) Greene, J.; Kahn, S.; Savoj, H.; Sprague, P.; Teig, S. Chemical Functional Queries for 3D Database Search. *J. Chem. Inf. Comput. Sci.* **1994**, *34,* 1297.

(20) Beroza, P. Combinatorial Library Design Using Three Dimensional Descriptors. American Chemical Society Meeting, Spring 1999.

(21) Bohm, H. J. *J. Comput.-Aided Mol. Des.* **1992**, *6*, 61.

(22) Zheng, W. F.; Cho, S. J.; Tropsha, A. Rational combinatorial library design. 1. Focus-2D: A new approach to the design of targeted combinatorial chemical libraries. *J. Chem. Inf. Comput. Sci.* **1998**, *38,* 251.

(23) Gillet, V. J.; Willet, P.; Bradshaw, J.; Green, D. V. S. Selecting Combinatorial Libraries To Optimize Diversity and Physical Properties. *J. Chem. Inf Comput. Sci.* **1999**, *39,* 169.

(24) Lewell, X. Q.; Judd, D. B.; Watson, S. P.; Hann, M. M. RECAP-Retrosynthetic combinatorial analysis procedure: A powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. *J. Chem. Inf. Comput. Sci.* **1998**, *38,* 511.

(25) Bures, M. G.; Martin, Y. C. Computational methods in molecular diversity and combinatorial chemistry. *Curr. Opin. Chem. Biol.* **1998**, *2,* 376.

(26) Bollobas, B. *Modern Graph Theory*; Springer Verlag: New York, 1998.

(27) Papadimitriou, C. H. *Computational Complexity*; Addison Wesley: New York, 1994.

(28) Garey, M. R.; Johnson, D. S. *Computers and Intractability: A Guide to the Theory of Np-Completeness*; Freeman: New York, 1979.

(29) Gianpaolo, B.; Bailey, N.; Leach, A.; Pozzan, A.; Green, D.; Hann, M. *Optimisation of 3D Focussed Libraries with PLUMS*: Noordwijkerhout, The Netherlands, 1999.

(30) Zhou, J.; Gu, X.; Cohen, J.; Stanton, R. V.; Eksterowicz, J.; Kassel, D.; Tarby, C. M.; Gubernator, K.; Meyers, P. *High-throughput Synthesis and Analysis of a Computationally Designed Dihydroisoquinolinone Library*. ACS National Meeting, New Orleans, 1999.

(31) Gillet, V. J.; Willett, P.; Bradshaw, J. The Effectiveness of Reactant Pods for Generating Structurally-Diverse Combinatorial Libraries. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 731−740.

CI990183C