Luddite: An Information-Theoretic Library Design Tool

Jennifer L. Miller,*,†,§ Erin K. Bradley,†,⊥ and Steven L. Teig‡

Deltagen, Inc., 740 Bay Road, Redwood City, California 94063, and Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, California 95134

Received July 18, 2002

We present an algorithm for the design of either combinatorial or discrete informative libraries. This approach is based on information theoretic techniques used extensively in coding theory. We have extended the information theoretic formalism to include an arbitrary number of property distribution constraints, such as Lipinski "drug-like" distributions. The method is demonstrated by comparing and contrasting a variety of different libraries selected from a single combinatorial source pool of compounds.

INTRODUCTION

The numerous genomics and proteomics efforts will certainly impact drug discovery by providing hundreds, if not many thousands, of novel targets. Nevertheless, each discovery project must still go through the process of finding drugs for these targets. This means that compounds must be synthesized and screened. Exactly which compounds to screen and at what stage to screen them are questions that have been debated for many years. Early discussions of QSAR[1] and Topliss' method[2] are particularly notable in this regard. Recently, combinatorial and parallel synthesis techniques have only added to this debate, transforming the question of which single compounds to assay to one of which *collection* of compounds to assay.

A collection of compounds is commonly referred to as a "library". There are many library design techniques that have been proposed, and they have been reviewed extensively.[3−9] Practitioners must decide whether to rely on compound properties or monomer properties in their design procedures.[10,11] They must also choose whether to select compounds that are diverse,[7,12−14] D-optimal,[15] informative,[16] or even more complex.[17] These design techniques can be amended to include other kinds of constraints that appeal to medicinal chemistry intuition. Most often, this means that compounds in designed libraries are also "drug-like"[18] by design,[19,20] or by filtering.[21−23] Most efforts hope to get "hits" from these libraries, which they then seek to convert to "leads" through a process referred to as analoging.

Here we will describe an algorithm for performing a library design strategy called *informative library design.*[16] The goal of informative design is to use molecules to "interrogate" the target receptor about what chemical features are required for binding. Each molecule, given its assortment of chemical features and its conformational flexibility, is able to ask many questions. The objective of informative design is to compose the library in such a way that a maximum number of conclusions can be drawn from the "answers" across all possible experimental outcomes (assay results). Mathematically, this optimization for maximum information can be accomplished by calculating and maximizing the Shannon entropy[24] for the outcomes tested by the library.[25] The initial application of Shannon entropy to the drug discovery field was in the HypoGen module of the Catalyst package.[26] More recently, it has been used to assess the degree of similarity between compound databases and to study the distribution of molecular descriptors within a database.[27−29]

Since informative design typically involves both model and data generation it can often be done best in the context of an iterative design-synthesis-screening cycle. After several rounds of going through this cycle the model for activity (i.e., which features are required for binding) tends to converge and is ready for application: e.g. designing combinatorial libraries or selecting compounds for testing from other sources.

Discrete libraries are defined as a collection of individual compounds, i.e., cherry-picks, and combinatorial libraries are defined as collections of compounds to be synthesized in a parallel or combinatorial fashion. The latter are often referred to as "matrix" libraries because of the array formats used to automate their synthesis (e.g. 96- and 384-well plates).[30−32] In this report we present algorithms for performing both discrete and combinatorial informative library designs. Then, we compare and contrast the results from a series of informative library designs ranging from a completely unconstrained discrete design to a combinatorial design subject to Lipinski Rule-of-5 type constraints.[18]

BACKGROUND AND METHODS

Both the motivation for designing informative libraries and a basic introduction to this method have been presented previously.[16,25,33] In particular, Teig[16] contains an early description of informative design as well as a qualitative comparison to other library design techniques such as synthesis-based, diversity, and coverage. Here, we take as our goal the design of the library that will allow us to discover the most information about our optimization target (e.g., the pharmacophores required for a high-affinity ligand). This manuscript presents two related strategies to produce

* Corresponding author phone: (415)490-2423; e-mail: jmiller@signaturebio.com.

† Deltagen, Inc.

‡ Cadence Design Systems, Inc.

§ Current address: Signature BioScience, Inc. 475 Brannan Street, San Francisco, California 94107.

⊥ Current address: Sunesis Pharmaceuticals, Inc., 341 Oyster Point Boulevard, South San Francisco, CA 94080.

10.1021/ci025566g CCC: $25.00
Published on Web 01/09/2003

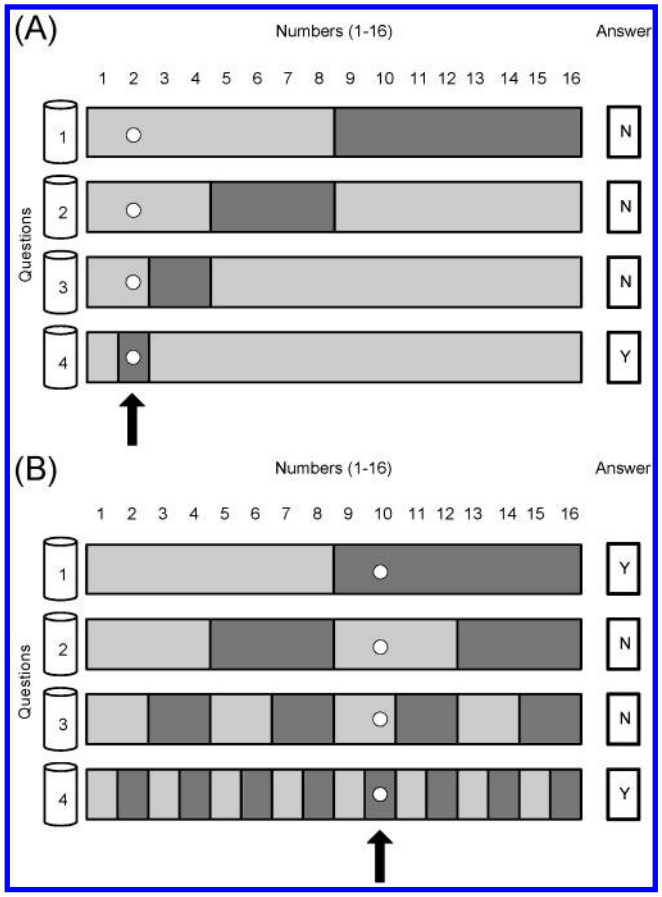

Figure 1. (A) A binary search strategy used to discover the correct number, (2), indicated by the arrow. The questions are shown as dark gray rectangles (e.g., Is the number greater than eight?). Each question is asked and answered in turn, allowing the player to customize the next question. (B). An informative search strategy used to discover the correct number, (10), indicated by the arrow. All four questions are asked at the same time, before any answers have been given. The questions shown form a general solution to the problem and can be reused each time the game is played.

these *informer* libraries. We begin with an intuitive explanation of the technique followed by the algorithmic details. Finally, we present comparative results among different discrete and combinatorial library designs.

Informative Design. The method of informative library design can be illustrated using a variant of the well-known children's game of 20 questions. In this illustration, the goal is to discover a number in the range 1−16 using a minimum number of yes-or-no questions. Obviously, the most efficient strategy is a binary search where the questions take the following form, "Is the number greater than eight?" It is easily demonstrated that four questions are always sufficient to determine the correct answer (Figure 1a). After each answer, the search area has been cut in half, and the next most efficient question can be formulated. The game becomes more challenging if the player is required to ask all of the questions at the same time. The strategy used in Figure 1a is no longer adequate unless the number is either 1 or 2. What is desired is the smallest number of questions that allows the player to determine the correct number without any intermediate answers. Figure 1b shows one such strategy. In this approach, each of the four questions tests different but overlapping dimensions of the search (hypothesis) space. By examining the pattern of answers to the questions, the player is able to discover the correct number in all cases. For example, if the pattern of answers is {Y, N, N, Y} the number must be 10. We will refer to this pattern of answers as a *codeword*—a term that comes from an analogous problem in coding theory: decoding a message transmitted over a noisy channel.[34] One key to the effectiveness of the proposed strategy is that every possible outcome corresponds to a single codeword. A second one is that the optimal set of questions can be asked simultaneously. A third key, and perhaps the most important one, is that the same set of optimal questions can be used every time the game is played.

This simple illustration is analogous to the library design problem. Instead of inferring the correct number, the goal becomes the discovery of the features required for ligand binding, desired phenotype and/or good pharmacokinetic properties. The range of possible answers is typically defined by the property that one is trying to optimize and how much information is available about the system of interest. For example, when optimizing ligand activity against an enzyme, the range of possible answers could include all four-point pharmacophores of a small descriptor set.[35] If the structure of the enzyme is known, this set of pharmacophores can be pruned using knowledge about the enzyme's active site.[36] In general, the set of possible answers should contain all those for which the prior probability is nonnegligible. For the purposes of this paper, "feature" is defined as a four-point pharmacophore.

The analogy extends to the questions as well. Our questions are represented as compound signatures, or fingerprints: an ordered vector of ones and zeros defining which features a compound contains.[37] As in the illustration, it is inefficient to screen one compound at a time in order to learn which features the target requires. What is desired is a set of compounds that are capable of interrogating the target to determine the features that are required for binding. Also as in the illustration, the most efficient set of compounds will test the target in different, but overlapping, ways. The pattern of assay results (answers) allows for the discovery of the binding features that are important to the target of interest. It should be noted that the inactive compounds play critical roles—just as "no" answers in 20 questions do—in discovering the relevant features. Once this set of features, or "model", has been discovered it can be used to select chemically diverse lead structures to be followed up in optimization efforts. It is critical for discovery efforts to find multiple, chemically diverse leads because of the failure rate of compounds due to pharmacokinetic problems.

While the above analogy is useful, it assumes a number of things that a library design approach cannot: for example, that every compound tests half of the possible features, that we can synthesize any compound in the design space, that every assay value is accurate, and that the goal is a single feature. Remarkably, the first three differences can be addressed by a slight refinement of the strategy using the mathematical techniques of coding theory.[34] The last difference can be addressed in various ways, one of which we discuss below.

From information theory,[24,33] we know that $\log_2(F)$ bits are required to decode F distinct outcomes. Analogously, as we can see from the illustration if each compound signature can cover half of the space then only $\log_2(F)$ compounds are required for an efficient design, where F is the number

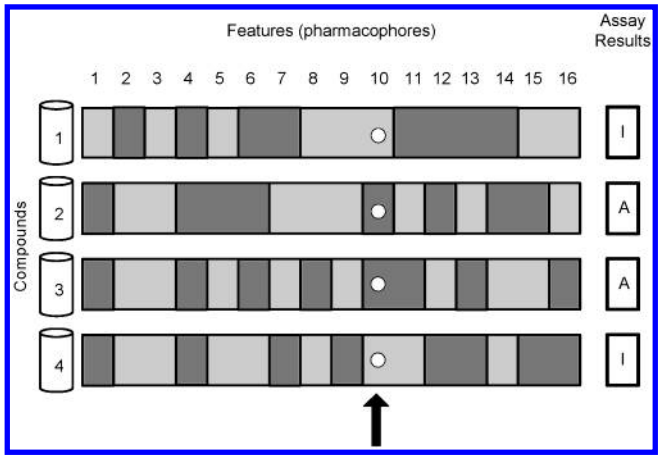

Figure 2. One possible informative library design to decode 16 features (pharmacophores). The compounds become the questions and the answers correspond to assay results (I = inactive, A = active). In this example, after decoding, we find that pharmacophore 10 is consistent with our assay results. This is one member of a large number of equally informative designs.

of features, or hypotheses. If this is not possible, as is the case when there are on the order of 10^7 features, more compounds are required to obtain unique codewords for each feature. In other words, each of the optimal $\log_2(F)$ questions can be seen as a single question composed from the fingerprints of many compounds. Accordingly, a loose upper bound on the number of compounds required to obtain unique codewords is given by

$$C_{UB} = \frac{F \log_2(F)}{2\bar{S}} \quad (1)$$

where C_{UB} is the number of compounds, F is the number of features, and $\bar{S}$ is the average number of 1's (i.e. pharmacophore bits) in a fingerprint. In our experience, the average number of pharmacophore bits per Universal Informer Library[33] (UIL) compound is ~25 000. The UIL is a general screening library of approximately 12 000 compounds. With 10^7 features, the informer library upper bound is slightly fewer than 5000 compounds. This number assumes that the process is error-free, but even 3× that number is only 15 000 compounds. This analysis does not take into account that a single compound can contribute to many questions.

To examine the second difference, we see that the ability of the set of questions to decode the message is invariant to column reordering. Thus, the set of four questions shown is just one of many equivalently good sets. In fact, unique decodability of outcomes is possible with any of the $F!$ or 2.1×10^{13} different sets. Therefore, it is not necessary that every compound in the design space be obtainable in order to find a maximally efficient set of questions. Figure 2 shows one possibility for an informative design.

As for the assumption that every assay result is correct, we turn to coding theory for insight. Specifically, we draw on the field of error-correcting codes (ECCs). Briefly, ECCs are sets of unique codewords that have a minimum pairwise Hamming distance (d) and, thus, are capable of correcting for (d-1)/2 errors in the transmission of a codeword.[38] With this distance between the codewords, the decoding process is robust to errors and becomes a simple nearest neighbor identification. In the pharmaceutical domain, the assay results

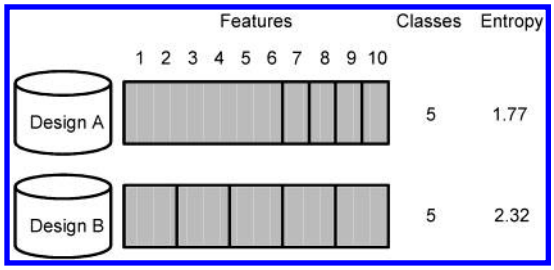

Figure 3. Two possible library designs to decode 10 features. Both designs have five feature classes but with different distributions across class size. For example, design A has one feature class of size 6 and four classes of size 1. The entropy of the distribution allows us to distinguish between the two designs. Design B is higher entropy and is the preferable library.

are analogous to the transmitted codeword. Unfortunately, while we can make conservative estimates, it is nearly impossible to assess, a priori, the error rate of the synthesis, purification, and assay. Nevertheless, the insight gained from examining this field is that, in addition to uniqueness, as long as we have some distance between our codewords, we gain robustness to errors in the experimental process.

The last assumption, while not explicitly addressed here, is overcome by approaching the discovery of the relevant set of features as an iterative process. Each round of (informative library design → compound synthesis → biological assay → data analysis) adjusts the probability that each feature is relevant. The process will provably converge if one prunes features with low probability during each round. It has been demonstrated experimentally that convergence to a selective and predictive ensemble of pharmacophores can be achieved in a small number of rounds.[35,39] Because the minimum number of compounds is used to test the feature set in each round, this method requires far fewer compounds than are typically used in screening libraries. This approach gives discovery efforts improvements in both efficiency (because far fewer compounds are screened) and probability of success (because the predictive ensemble can be used to identify topologically different lead compounds).

To summarize this approach, our goal is to determine the set of features that predict activity against a particular target. Our input is the universe of features that are conceivably relevant and a set of compounds that we can use to interrogate the target. In this context, a binary signature represents the set of features that a compound can test. We are seeking to select a subset of these compounds that will interrogate the target in different but slightly redundant ways. Specifically, our primary focus is not on the compounds themselves but on the unique codewords (i.e., pharmacophores) a given set of compounds contains. The next sections state this goal quantitatively and describe the algorithms used to select an informative library.

Cost Function. Given a set of features, we seek a set of compounds that will allow us to decode each individual feature (Figure 2). In the event this is not possible, we seek to decode as many features as possible, with the flattest distribution across the size of feature classes (Figure 3). A feature class refers to a subset of features that all have the same codeword. Two examples of this situation are shown in Figure 3. While a simple count of the number of decodable features is not sufficient to choose between the cases in Figure 3, the entropy of the class distribution is well suited

to accomplish this task. Entropy, in an information-theoretic sense, is a measure of uncertainty. If, in the extreme case, all of the codewords are the same (i.e. minimum entropy of the feature classes), then there is no uncertainty about the outcome, and we gain no additional information upon assaying the compounds. At the other extreme, all of the codewords are different (i.e. maximum entropy of the feature classes), there is great uncertainty about the outcome and we gain a lot of information. Looked at another way, a uniform distribution across feature class sizes reduces the expected effort in follow-up experiments. Thus, we try to maximize the cost function, or measure, given by

$$M = H = -\sum_{i=1}^{C} \frac{||c_i||}{F} \ln \frac{||c_i||}{F} \qquad (2)$$

where M is the library measure, H is the entropy of the feature classes, C is the number of distinct classes, F is the number of features in the design space, and $||c_i||$ is the size of feature class i. During the course of the optimization, we seek a set of compounds that maximizes H.[25,40]

It is often desirable to constrain a library selection to satisfy certain physical property distributions. For example, we might want to tailor a library so that the molecules are "drug-like". One of the ways to accomplish this is to prefilter the source pool with Lipinski's Rule-of-5.[18] Another way—one that is more reflective of Lipinski's findings—is to require that the set of molecules in the selected library satisfy the various property *distributions* identified in his study. To accommodate this type of constraint, we extended the cost function to include a variable number of terms as follows:

$$D = \sum_{j=1}^{p} \omega_j \sum_{k=1}^{b} (\rho_d - \rho_c)^2 \qquad (3)$$

Here, D is the total cost of the distributions. The outer summation is over the p property distributions included in the optimization. Each distribution is scaled by a weight term, ω, which allows the user to control the relative importance of the various optimization constraints. The inner summation is calculated for each property distribution and contains a cost per bin: the square of the difference between the desired fraction ρ_d and the current fraction ρ_c. There are other functions to calculate the distance between two discrete distributions (e.g., Kullback-Leibler[34]), but this simple function is easy to calculate and has desirable characteristics. It penalizes bins that are too empty and too full, and its value is zero when the distribution is matched. The initial implementation of this method did not penalize bins that were too full and was not sufficient to obtain the desired property distributions.[41] A library optimized under these distribution constraints maximizes $M = H - D$. Therefore, a set of compounds that matches the desired distributions will result in no added cost to the state measure.

Discrete and Combinatorial Algorithms. Beginning with a list of compounds we could synthesize (source pool), our goal is to select a subset of the list that maximizes the measure, H. We use the term "state" to refer to a set of compounds for which we can calculate the measure. From eq 2, we note that the measure of a state is a function of the classes, but our moves through state space are a function of the compounds [rows]. In general, our measure cannot be calculated incrementally and must be completely reevaluated whenever the state changes. This situation is in stark contrast to other library design methods where the measure can be updated as a function of only those compounds that have changed. Despite this seeming limitation, the method is very efficient.

```
Select the highest entropy compound in the source pool
For i ← 2 to N do
    For j ← 1 to M do
        Add component j and calculate new state measure
        If (new state measure > best state measure) then do
            Best state ← new state
        End if
    End j
End i
While (optimize == true) do
For i ← 1 to N do
    Subtract component i from Best state
    For j ← 1 to M do
        Add component j and calculate new state measure
        If (new state measure > best state measure) then
            Best state ← new state
        Endif
    End j
End i
If (Best state wasn't reset) then
    Optimize = false
Endif
End while
```

Figure 4. Pseudocode for the 2-pass discrete library optimization. N refers to the desired number of compounds. M refers to the number of compounds in the source pool. This algorithm forms the basis for the combinatorial case.

At the initiation of a therapeutic project the number of features that we typically attempt to decode is on the order of 10^7. This number represents all 4-point pharmacophores possible using six descriptors and seven interdescriptor distance bins. For this work, we used a very standard descriptor set: hydrogen-bond donors and acceptors, positive and negative ionizable groups, hydrophobic groups, and aromatic ring centroids. Given source pool sizes that range from tens of thousands into the millions of compounds, the algorithms presented in this section attempt to strike a balance between memory usage and calculation speed. We complete the algorithms section with a description of the program options that are unique to each type of library design.

The algorithms for the discrete and combinatorial designs are very similar. They differ in how the source pool is sampled and in how the final library is selected. For the purposes of this description, "components" refers to compounds in the discrete library and monomers in the combinatorial library. As shown in the pseudocode in Figure 4, the basic approach to either type of optimization is a greedy build-up[42] of the library to the desired number of components, followed by a second phase that reevaluates each of the library components to see if a better selection is available. If a better choice can be made, the components are exchanged, and the process of reevaluation begins again. This second phase continues until no improvement in the components is possible.

In the discrete algorithm, components ≡ compounds. By default, all of the compounds are sampled during each greedy pass. This is true in both the selection and optimization phases. We begin by selecting a seed compound: the highest

```
Select the highest entropy compound in the source pool
Add all of the compound's monomers to the library
While (dimensions not full)
    For i ← 1 to D do
        For j ← 1 to M_i do
            Find all compounds in hyperplane j
            Add hyperplane j and calculate new state measure
            If (new state measure > best state measure) then do
                Best state ← new state
                Add new monomers from hyperplane j
                If (any dimension is full) then do
                    Retire the dimension
                End if
            End if
        End j
    End i
End while
While (optimize == true) do
    For i ← 1 to D do
        For j ← 1 to M_i do
            Subtract monomer j from Best state
            For k ← 1 to M_i do
                Find all compounds in hyperplane k
                Add hyperplane k and calculate new state measure
                If (new state measure > best state measure) then
                    Best state ← new state
                Endif
            End k
        End j
    End I
    If (Best state wasn't reset) then
        Optimize = false
    Endif
End while
```

Figure 5. Pseudocode for the 2-pass combinatorial library optimization. *D* refers to the number of dimensions or monomers. *M* refers to the number of monomers in a given dimension.

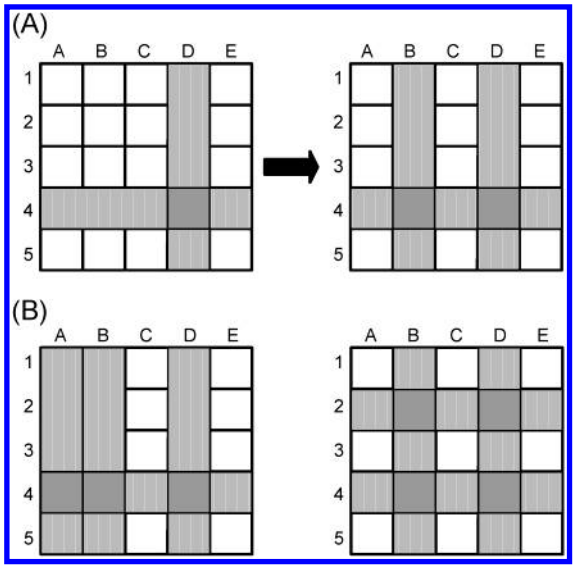

Figure 6. (A) Example of the first two steps for a combinatorial library selection. This example shows a two-component reaction with a source pool that contains five monomers at each substituent position. The first step is to select the highest entropy compound in the source pool. This is shown on the left as compound **D4**. Monomers **D** and **4** define a move set (shown in light gray) for the next compound selection. As shown in the figure, compound **B4** is the best selection from the move set. The library grows as a matrix by adding the best hyperplane from the current move set. (B). Two possibilities for the third hyperplane selection. The move on the left shows a single compound (**A4**) and on the right two compounds (**B2, D2**). Both moves are part of the move set (gray rows/columns) and either can be selected as the next best move.

entropy compound in the source pool. The remainder of the first pass consists of iteratively selecting the next best compound in terms of maximizing the state measure. This process is continued until either the requested number of compounds has been selected or the optimization is complete. This allows the program to stop when the measure has reached an optimal value, or it has reached a local maximum. An optional second pass follows that optimizes this set by allowing each compound to be swapped for a more informative selection.

In the combinatorial algorithm, components ≡ monomers (Figure 5). The final library must be the set of monomers from each dimension such that the cross products [compounds] maximize the state measure. This additional constraint influences how we can sample the source pool. Instead of searching through individual compounds in the source pool, we search through and select hyperplanes. A hyperplane is an *n-1* dimensional slice of an *n*-dimensional plane. For the purposes of a combinatorial library, *n* equals the number of substituent, or monomer, positions. By restricting our moves to hyperplanes, we guarantee that our selected library will be a matrix. As in the discrete case, we begin by selecting the highest entropy compound. This selection implies the selection of one monomer from each dimension (Figure 6a). From here, all of the relevant hyperplanes are sampled during each pass of the library selection and optimization. During the build-up phase, moves are made in "matrix" space to allow the optimization to stop at any time and still provide a dense matrix. This algorithm is briefly explained below in the form of a two-dimensional example. The algorithm generalizes to any number of dimensions.

As mentioned above, a hyperplane is defined as an *n-1* dimensional slice. For $n = 1$, this is a point; for $n = 2$, this is a line; for $n = 3$, this is a flat surface; etc. An example of this type of move selection is shown in Figure 6a. On the left is a two-dimensional supermatrix (source pool) from our library is to be selected. The pick is seeded with a single compound (**4D**), shown in dark gray. Because this is a greedy algorithm the selection of a seed compound determines two of the final monomers: **4** and **D**. To select the next best hyperplane, the set of moves to evaluate is shown in light gray. It consists of all of the compounds in row **4** plus all of the compounds in column **D**. As shown, the best possible move is to select compound **4B**. The move set now contains an additional column **B**. The set of potential moves now consists of two types of hyperplanes: single compounds and pairs of compounds. For example, shown in Figure 6b are two moves from the move set. The one on the left contains compound **4A**. The one on the right contains two compounds **2B** and **2D**. Both moves are valid in terms of maintaining a matrix during the build-up phase, but the hyperplanes contain different numbers of compounds. The hyperplane on the right is most likely the better move from the point of view of maximizing entropy.

This process is continued until either the requested number of monomers in each dimension has been selected or the optimization has completed. As in the discrete case, this allows the optimization to stop when the measure has reached an optimal value, or it has reached a local maximum. The combinatorial library can be further optimized using the same process as in the discrete optimization. Again, all component choices are reevaluated. This means that during each optimization pass, every monomer is dropped, the rest of the library is used, and all possible monomer replacements are evaluated. This evaluation is done for all of the

Table 1. Results of Example A + B + C → D Discrete and Combinatorial Library Designs

compd set[a]	properties used[b]	subsample size	second pass[c]	measure	entropy[d]	classes[e]	features not tested
ideal					16.23	77 075	0
source pool					16.22	76 786	158
D1	0			16.16	16.16	75 365	282
D2	0	5000	N	16.16	16.16	75 339	303
D3	0	5000	Y	16.16	16.16	75 339	303
D4	4	5000	Y	16.04	16.10	74 395	508
D5	4	5000	N	16.04	16.10	74 395	508
C1	0		N	15.85	15.85	71 633	1804
C2	0		Y	15.90	15.90	72 053	1537
C3	4		N	15.61	15.82	71 491	2000
C4	4		Y	15.69	15.83	71 549	1905
R (av)				14.40	14.63	58 699	6294

[a] Sets beginning with "D" refer to discrete designs, with "C" refer to combinatorial designs. *R*(av) is an average of 10 random selections. [b] Refers to the number of R5 properties used to constrain the optimization. [c] Refers to whether the greedy selection was further optimized. [d] Entropy is calculated from $H = -\sum_{i=1}^{c} ||c_i||/F \ln ||c_i||/F$ as described in the text. [e] Refers to the number of distinct feature classes, $||C_i||$, used to calculate the entropy.

dimensions. If a better monomer selection is possible, *in any dimension*, it is made, and the process begins again.

Extensions to the Basic Algorithm. It is often desirable to specify that certain compounds or monomers be included in a library. For example, we may desire to incorporate literature compounds, "medicinal chemistry intuition" about fragments and/or substructures, or other pragmatic compound choices into our final design. The algorithms shown in Figures 4 and 5 can be easily extended to include this ability. We have implemented an option in our program that allows the user to define the compounds, or monomers, that must be a part of the final library. The algorithms for both the build-up and optimization passes are aware of these requirements and optimize around them. This is very similar to the "hole-filling" problem encountered in diversity designs.

We included two other options in the discrete algorithm. The first is the ability to subsample the source pool during both the build-up and optimization phases. With this option, instead of examining the entire source pool to find the best compound, the program will choose a set of compounds at random and select the best one from that set. The user specifies the size of this sample. Additionally, because there is a chance that none of the compounds from the subsample will increase the measure, the user also specifies how many times this subsample should be drawn from the source pool before giving up. This dramatically decreases the run time of the program while only slightly impacting the quality of the designs. This strategy can be easily extended to a parallel environment where individual processors could evaluate different random subsamples of the source pool, thereby increasing the overall sampling during each pass.

The second of these options is the ability to define the minimum Tanimoto fingerprint similarity between any two compounds in the discrete library. For example, when using pharmacophore fingerprints, a user may require that the library maximize entropy at the same time as minimize overlap between the compound fingerprints. It has been shown that, while diverse libraries are not informative, informative libraries are generally diverse.[25] It is not strictly necessary that informative libraries are also diverse, but sometimes it is desirable. This option allows the user to specify that an intersection of informative and diverse be selected.

All algorithms were implemented in the C++ programming language under Microsoft Windows NT. Calculations were run on workstations equipped with a 400 MHz Intel Pentium III and 500 MB of RAM.

RESULTS AND DISCUSSION

In this section we present the results of nine different libraries selected using this algorithm. We assess the quality of these libraries by comparing the final measures to both the optimal result and to a random result. As outlined in a previous section, the optimal result is one in which all features are decodable by the library of compounds.

All libraries were selected from the same 273 373 compound source pool. This source pool comes from a three-component reaction, A + B + C → D, with monomer lists of length 33, 436, and 19. 4-point pharmacophore signatures were calculated for all compounds in the source pool.[37] Because this example came from an active therapeutic project, the number of hypotheses under consideration was only 77 075. For the purposes of comparison, all libraries were selected to contain 250 compounds. For the combinatorial libraries, this total was achieved by selecting a matrix of dimensions 5 × 10 × 5.

Table 1 contains the results of the different library designs. The five discrete (D1-D6) and four combinatorial (C1−C4) designs allow us to compare and contrast the different library types and use of program options. In addition, Table 1 contains the average results from 10 random picks of 250 compounds from the source pool. For all libraries, Table 1 contains the options used to do the calculations as well as the results in terms of the final measure, entropy, and details about the number of classes and features not tested. The first two rows of Table 1 contain the ideal results for this feature space and the best results possible using this source pool. We see that this source pool is not capable of decoding all of the features in the space. This is a common problem with combinatorial source pools and reflects the real-life nature of these examples. For this source pool, our "optimal" result is one where the 76 786 features are decoded. On average, the random trials are capable of decoding only 76% of those features.

The first discrete library design (D1) is used as a baseline for comparison. This calculation represents the "best pos-

sible" result under the algorithms presented in the previous section. This library of only 250 compounds is capable of decoding 98% of the feature classes that the entire source pool can decode. It does a much better job than a random selection of compounds (R). However, this is the most computationally expensive of the algorithmic choices. For efficiency, the four other discrete designs were done using a subsample of 5000 compounds during each compound selection and optimization pass. These four libraries were generated with and without the second pass optimization and the Rule-of-5 (R5) constraints.[18] The weight values for the R5 constraints were determined through a process of trial and error. They have been shown to provide a good balance between entropy and R5 properties for this source pool.[43]

As Table 1 shows, there was only a slight decrease in the number of classes with the subsampling of the source pool (D2). We also observe that in the absence of any additional constraints, the computationally expensive second pass optimization does not improve the quality of the selection (D2 vs D3). This was also true for the libraries selected with and without the R5 constraints (D4 vs D5). All four libraries compared well with the optimal result. There was only a 1% loss of decodable classes and a slight decrease in the final library measure when the R5 constraints were included (D2 vs D5). This is expected because we are optimizing two competing forces. While it can be hard to intuitively understand the decrease in "entropy", this difference is due to the loss of 944 decodable classes and the gain in the number of features that are not tested at all.

Unlike the discrete cases, we do see an effect from the second pass optimization in the combinatorial libraries. This is true for both the unconstrained and R5 constrained designs (C2 and C4). The increase in the measure is small, however, and may be considered negligible. Without R5 constraints, the second pass only improved the number of decodable feature classes by 420 (0.6%). With them, the improvement was only 58 total classes (0.1%). In all cases we found that a variable number of optimization passes was required for convergence. It is not surprising that the combinatorial libraries required optimization after the initial build-up, or selection, phase because the use of hyperplanes to search the source pool is a very coarse-grained approach compared with the single compound search used in the discrete designs. We also see an effect from adding the R5 constraints to the optimization. Comparing the results from C2 and C4, there was a loss of 504 classes (0.7%). The best performing combinatorial library (C2) compared with the best discrete library (D1) shows that we are making compromises in the measure in order to gain cost-effectiveness and simplicity of synthesis. However, even in the presence of matrix constraints, the combinatorial optimization algorithm is vastly superior to a random selection.

Another interesting comparison between the discrete and combinatorial algorithms is to look at the entropy of the libraries as a function of library size during the greedy phase (Figure 7). Each step in the greedy phase represents the best selection for that number of compounds. Both curves rise rapidly as a function of library size with the combinatorial algorithm lagging behind the discrete one for performance. We see from Figure 7 that a discrete library of only 91 compounds has the same measure as the optimal combinatorial library of 250 compounds. While this is a much smaller

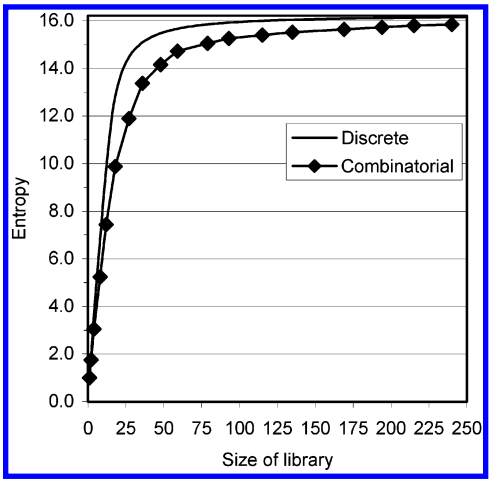

Figure 7. Typical results for a discrete (black line) and combinatorial (black line with diamonds) build-up in an unconstrained optimization. This graph shows the library entropy vs size (number of compounds). The discrete library has a more rapid increase in entropy but converges to approximately the same result as the combinatorial library. The diamonds indicate the results after the selection of each optimal hyperplane.

library for the same degree of decodability, it still might be most cost-effective to synthesize the combinatorial library. As a general rule, twice as many compounds are required in a combinatorial library in order to achieve the same information content in a discrete library. In an iterative setting, combinatorial libraries are desirable in the early rounds of discovery, with discrete libraries playing an important role in later rounds when the ability to cherry-pick specific compounds becomes critical for model refinement.

CONCLUSIONS

In this paper we have described an efficient algorithm to select an informative set of compounds from a source pool. The method is capable of selecting and optimizing both discrete and combinatorial libraries. Additionally, the method has been implemented to allow users to constrain the optimization by forcing certain compounds or monomers into the final library as well as by skewing the library toward certain property distributions.

Our goal is to identify a set of compounds that, upon assay, are able to disambiguate a given set of features. As in an earlier implementation of this method,[25] we have used the entropy of the feature classes decoded by the compounds as one part of our cost function. We have supplemented the entropy with an optional penalty term used when optimizing in the presence of desired property distributions for the set of compounds. The number of features that our algorithm can address is not restricted and has been used to optimize libraries against a few hundred features up to ~30 million. Additionally, the combinatorial adaptation of the discrete algorithm is capable of generalizing to any number of dimensions (i.e., any number of components in the reaction).

One of the more important implications of our formulation is that the cost function enables us to compare library designs quantitatively. The comparison can be absolute (e.g., how far from ideal is this library?) or relative (e.g., is the discrete or combinatorial subset of this source pool more informative?). This is a nontrivial task for other methods,[13,15,44,45] but a result of the optimization process for this method. We

have used these types of evaluations to help project teams choose which template to pursue. We have also used this comparative approach to determine when it is necessary (to gain information) to shift from making combinatorial libraries to discrete ones. Additionally, our project teams have used this tool to design combinatorial libraries and then to fill the information gaps with discrete compounds.

Finally, we have included a small set of library selections from the same source pool in order to assess our algorithm's performance. We have shown that all results are much better than random picks of the same size and only slightly worse than the theoretically best result for the source pool. While the discrete libraries were universally able to decode more feature classes than the combinatorial libraries, the combinatorial sets were not strikingly worse and represent much less expensive libraries to synthesize. On the other hand, we also found that a small number of discretely selected compounds are equivalent to a much larger set of combinatorially selected compounds.

ACKNOWLEDGMENT

J.L.M. would like to thank Randy Henne for many helpful discussions and Rob Stanton and Julie Penzotti for testing the program.

Note Added after ASAP Posting. This article was released ASAP on 1/9/2003 with an incorrect affiliation for two authors and a current address missing for an author. The correct version was posted on 1/13/2003.

REFERENCES AND NOTES

(1) Hansch, C.; Fujita, T. r-s-p analysis − A method for the correlation of biological activity and chemical structure. *J. Am. Chem. Soc.* **1964**, *86*, 1616−1626.
(2) Topliss, J. G. Utilization of operational schemes for analogue synthesis in drug design. *J. Med. Chem.* **1972**, *15*, 1006−1011.
(3) Beno, B. R.; Mason, J. S. The Design of Combinatorial Libraries Using Properties and 3D Pharmacophore Fingerprints. *Drug Discov. Today* **2001**, *6*, 251−258.
(4) Blaney, J. M.; Martin, E. J. Computational Approaches for Combinatorial Library Design and Molecular Diversity Analysis. *Curr. Opin. Chem. Biol.* **1997**, *1*, 54−59.
(5) Leach, A. R.; Hann, M. M. The In Silico World of Virtual Libraries. *Drug Discov. Today* **2000**, *5*, 326−336.
(6) Matter, H.; Baringhaus, K. H.; Naumann, T.; Klabunde, T.; B., P. Computational approaches towards the rational design of drug-like compound libraries. *Comb. Chem. High Throughput Screen* **2001**, *4*, 453−475.
(7) Pearlman, R. S.; Smith, K. M. Novel Software Tools for Chemical Diversity. *Perspect. Drug Discovery Des.* **1998**, *9*, 339−353.
(8) Spellmeyer, D. C.; Grootenhuis, P. D. J. Recent Developments in Molecular Diversity: Computational Approaches to Combinatorial Chemistry. *Annu. Rep. Med. Chem. Rev.* **1999**, *34*, 287−296.
(9) Willett, P. Chemoinformatics − Similarity and Diversity in Chemical Libraries. *Curr. Opin. Biotechnol.* **2000**, *11*, 85−88.
(10) Rose, S. Statistical design and application to combinatorial chemistry. *Drug Discov. Today* **2002**, *7*, 133−138.
(11) Martin, E. J.; Hoeffel, T. J. Oriented substituent pharmacophore PRopErtY space (OSPPREYS): a substituent-based calculation that describes combinatorial library products better than the corresponding product-based calculation. *J. Mol. Graph. Model.* **2000**, *18*, 383−403.
(12) Zheng, W.; Cho, S. J.; Waller, C. L.; Tropsha, A. Rational Combinatorial Library Design. 3. Simulated Annealing Guided Evaluation (SAGE) of Molecular Diversity: A Novel Computational Tool for Diverse Library Design and Database Mining. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 738−746.
(13) Schnur, D. Design And Diversity Analysis Of Large Combinatorial Libraries Using Cell-Based Methods. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 36−45.
(14) Brown, R. D.; Martin, Y. C. Designing Combinatorial Library Mixtures Using a Genetic Algorithm. *J. Med. Chem.* **1997**, *40*, 2304−2313.
(15) Martin, E. J.; Blaney, J. M.; Siani, M. A.; Spellmeyer, D. C.; Wong, A. K. et al. Measuring Diversity: Experimental Design of Combinatorial Libraries for Drug Discovery. *J. Med. Chem.* **1995**, *38*, 1431−1436.
(16) Teig, S. L. Informative libraries are more useful than diverse ones. *J. Biomol. Screening* **1998**, *3*.
(17) Agrafiotis, D. K. Multiobjective optimization of combinatorial libraries. *IBM J. Res. Dev.* **2001**, *45*, 545−566.
(18) Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings. *Adv. Drug Delivery Rev.* **1997**, *23*, 3−25.
(19) Brown, R. D.; Hassan, M.; Waldman, M. Combinatorial library design for diversity, cost efficiency, and drug-like character. *J. Mol. Graph. Model.* **2000**, *18*, 427−437.
(20) Martin, E. J.; Critchlow, R. E. Beyond Mere Diversity: Tailoring Combinatorial Libraries for Drug Discovery. *J. Comb. Chem.* **1999**, *1*, 32−45.
(21) Blake, J. F. Chemoinformatics - - Predicting the Physicochemical Properties of 'Drug-like' Molecules. *Curr. Opin. Biotechnol.* **2000**, *11*, 104−107.
(22) Clark, D. E.; Pickett, S. D. Computational Methods for the Prediction of 'Drug-likeness'. *Drug Discov. Today* **2000**, *5*, 49−58.
(23) Muegge, I.; Heald, S. L.; Brittelli, D. Simple Selection Criteria for Drug-Like Chemical Matter. *J. Med. Chem.* **2001**, *44*, 1841−1846.
(24) Shannon, C. E. A Mathematical Theory of Communication. *Bell System Tech. J.* **1948**, *27*, 379−423, 623−656.
(25) Barnum, D.; Greene, J.; Teig, S. L. Designing maximally informative libraries. ACS National Meeting, Dallas, TX, 1998.
(26) *Catalyst v.4.0*; Accelrys, Inc.: San Diego, CA.
(27) Stahura, F. L.; Godden, J. W.; Xue, L.; Bajorath, J. Distinguishing between natural products and synthetic molecules by descriptor Shannon entropy analysis and binary QSAR calculations. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 1245−1252.
(28) Godden, J. W.; Bajorath, J. Shannon entropy−a novel concept in molecular descriptor and diversity analysis. *J. Mol. Graph. Model.* **2000**, *18*, 73−76.
(29) Godden, J. W.; Stahura, F. L.; Bajorath, J. Variability of molecular descriptors in compound databases revealed by Shannon entropy calculations. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 796−800.
(30) Agrafiotis, D. K.; Lobanov, V. S. Ultrafast algorithm for designing focused combinatorial arrays. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 1030−1038.
(31) Pearlman, R. S.; Smith, K. M. Combinatorial library design in the new century. 221st ACS National Meeting, San Diego, CA, 2001.
(32) Stanton, R. V.; Mount, J.; Miller, J. L. Combinatorial Library Design: Maximizing Model-Fitting Compounds within Matrix Synthesis Constraints. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 701−705.
(33) Myers, P. L.; Greene, J. W.; Saunders: J.; Teig, S. L. Rapid, Reliable Drug Discovery. *Today's Chemist Work* **1997**, *6*, 46−48.
(34) Cover, T. A.; Thomas, J. A. *Elements of Information Theory*; John Wiley & Sons: New York, 1991; 1−49.
(35) Bradley, E. K. Application of Informative Library Design to the Selection of Chemical Libraries for Lead Generation and Optimization. *J. Med. Chem.* **2002**, To be submitted.
(36) Eksterowicz, J. E.; Evensen, E.; Lemmen, C.; Brady, G. P.; Lanctot, J. K. et al. Coupling Structure-Based Design with Combinatorial Chemistry: Application of Active Site Derived Pharmacophores with Informative Library Design. *J. Mol. Graph. Model.* **2002**, *20*, 469−477.
(37) Bradley, E. K.; Beroza, P.; Penzotti, J. E.; Grootenhuis, P. D. J.; Spellmeyer, D. C. et al. A rapid computational method for lead evolution: Description and Application to a1-Adrenergic Antagonists. *J. Med. Chem.* **2000**, *43*, 2770−2774.
(38) Cameron, P. J. *Combinatorics: Topics, techniques, algorithms*; Cambridge University Press: London, 1995.
(39) Miller, J. L. The Discovery Engine In Action: Demonstration of Rapid Lead Evolution. *Computational Drug Design*: San Francisco, CA, 2000.
(40) Greene, J.; Barnum, D.; Teig, S. L. unpublished results, 1998.
(41) Miller, J. L. unpublished results, 2000.
(42) Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. *Introduction to Algorithms*; MIT Press: Cambridge, 1998.
(43) Bradley, E. K. unpublished results, 2001.
(44) Agrafiotis, D. K. A constant time algorithm for estimating the diversity of large chemical libraries. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 159−167.
(45) Agrafiotis, D. K. On the use of information theory for assessing molecular diversity. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 576.

CI025566G