ARTICLES

GLARE: A New Approach for Filtering Large Reagent Lists in Combinatorial Library Design Using Product Properties

Jean-François Truchon* and Christopher I. Bayly

Merck Frosst Canada & Co., 16711 Trans Canada Hwy., Kirkland, Québec, Canada H9H 3L1

Received November 7, 2005

We present a novel computer algorithm, called GLARE (Global Library Assessment of REagents), that addresses the issue of optimal reagent selection in combinatorial library design. This program reduces or eliminates the time a medicinal chemist spends examining reagents which a priori cannot be part of a "good" library. Our approach takes the large reagent sets returned by standard chemical database queries and produces often considerably reduced reagent sets that are well-behaved with respect to a specific template. The pruning enforces "goodness" constraints such as the Lipinski rule of five on the product properties such that any reagent selection from the resulting sets produces only "good" products. The algorithm we implemented has three important features: (i) As opposed to genetic algorithms or other stochastic algorithms, GLARE uses a deterministic greedy procedure that smoothly filters out nonviable reagents. (ii) The pruning method can be biased to produce reagent sets with a balanced size, conserving proportionally more reagents in smaller sets. (iii) For very large combinatorial libraries, a partitioning scheme allows libraries as large as 10^{12} to be evaluated in 0.25 s on an IBM AMD Opteron processor. This algorithm is validated on a diverse set of 12 libraries. The results that we obtained show an excellent compliance to the product property requirements and very fast timings.

1. INTRODUCTION

The emergence of high-throughput technologies for the synthesis and screening of a large number of compounds has led to unprecedented possibilities for drug development. Although thousands of compounds can be synthesized rapidly with automation technology, scientists have realized that it is not sufficient to blindly generate many compounds for screening. Rather, a rational evaluation of the combinatorial library is desirable in order to maximize the outcome of an expensive synthesis campaign.[1,2] This can be achieved by paying careful attention to both the design of the synthetic scheme and the reagent selection. In this latter task, a great amount of choice is offered: for example, the Available Chemicals Directory (ACD)[3] provides chemists with a way to find a supplier for any of more than 186 000 chemical substances representing 1.25 million individual chemical products from over 600 suppliers. If a combinatorial library is made with two or more reagent functionalities, between 100 000 and many billions of different products are accessible. The ACD offers from 145 to about 6000 reactants per functional group, whereas around 20 need to be chosen in a typical case to carry forward the synthesis. Therefore, between 86% and 99.7% of the reagents available would need to be pruned from the original sets! Worse than that, many publications have shown that it is important to consider the products instead of each individual reagent for better results.[4−7] The idea that reagent selection is dependent upon the library template and should, therefore, be done according to the products seems intuitive. With millions or billions of candidate products, optimal selection becomes intractable by humans; computer-assisted selection becomes essential.

For high-throughput virtual screening, filters such as the Lipinski rule of five[8] have been used as well as some specific QSAR methods based on simple properties. When the library is made for screening against a specific target, docking and scoring approaches can be utilized, however, with a much lower throughput. If the objective of making the library is to enrich a screening compound repository, then diversity-based filters may be more appropriate. In all cases, there is a need to reduce the space of product possibilities. In this article, we report a new general approach for monomer selection based on product properties with unprecedented low human time and computer time needs.

Because we want to address the "high-throughput" end of the spectrum, we validated the algorithm with properties such as the number of hydrogen-bond acceptors (HBA), the number of hydrogen-bond donors (HBD), the number of non-hydrogen atoms (nonH), and the calculated log of the octanol−water partition (logP). Using nonH instead of the molecular weight has the advantage of being more tolerant of molecules containing bromine, chlorine, or sulfur atoms, which are frequently part of drugs and development candidates. Other properties could have been selected as well.

Computer programs for library design need to handle both the product properties calculation and the large number of

* Corresponding author. Phone: (514) 428-3144. Fax: (514) 428-4900. E-mail: jeanfrancois_truchon@merck.com.

10.1021/ci0504871 CCC: $33.50
Published on Web 05/26/2006

possible products. A lot of progress has been made in applying stochastic methods to the problem of reagent selection. They include optimization techniques such as genetic algorithms found in SELECT,[9] MOSELECT,[10] GALOPED,[11] VOLGA,[12,13] and the work of Sheridan et al.;[14] simulated annealing in SAGE[15] and PICCOLO;[16] Monte Carlo;[17,18] and fast exchange.[19] They are fairly general and can handle many useful product properties as well as sophisticated scoring schemes. However, stochastic algorithms have been applied only to quite small virtual libraries (less than 10 000 products). This is due, in part, to the need to build the chemical structure for combinatorial products (henceforth referred to as "enumeration"); this becomes computationally impractical for hundred of thousands or more compounds. These algorithms usually select a small set of optimal reagents that does not give much choice to the synthetic chemist, excepting MOSELECT, which produces several equally fit selections. These methods are so CPU-intensive that scientists need to first invest quite a lot of effort in pruning down the monomer sets obtained from a query to the ACD down to a tractable size. We suggest that our algorithm be applied in a first pass to eliminate undesired reagents from larger lists, greatly reducing the list size. Then, more sophisticated filtering rules in a lower-throughput mode can fine-tune the list.

Another class of algorithms, to which GLARE (Global Library Assessment of REagents) belongs, is based on deterministic procedures. As opposed to other paradigms where a best final synthetic solution is offered, these algorithms produce a set of reagents that is well-behaved but usually larger than what is really needed for the library. Furthermore, they do not rely on sampling but rather on a greedy iterative process. Results comparable to those of stochastic algorithms have been obtained.[20,21] The first types of algorithms in this direction are MFA[22] and DMFA,[21] where the selection is made among the top reagents that have been rank-ordered on the basis of their frequency in "good" products. Although very quick, these methods do not guarantee that the products formed by the selection are all "good". At the other extreme, algorithms such as PLUMS[21] are very careful at eliminating the worst reagent at each step in an iterative process. This requires a lot of computational work, mostly because only one reagent is removed at each step and many subarray scores need to be obtained. Stanton et al.[23] have also designed a "cut-down" iterative algorithm that enumerates all of the products and removes the reagents with the lowest occurrence in "good" products. They report optimizing a 1000×1000 library in 2.1 s on a Pentium 400 MHz machine. For higher throughput, very quick random sampling methods have also been designed,[24] and virtual libraries of 10^{10} products could be optimized. However, because of their crude way of eliminating reagents, very few compounds remain at the end. Most of these deterministic methods necessitate a full enumeration of the products, intractable for medium to large sets of reagents such as those typically obtained from a query to a chemical database.

In this article, we present a new deterministic reagent-filtering algorithm based on product properties, which avoids the formal enumeration of the products themselves. The resulting cleaned reagent sets lead to a library highly enriched in "good" products in a controllable fashion. We use an optional partitioning scheme similar to random sampling except that all of the reagents are scored in one combination step using pooling like in sports events. This improvement uses a pseudo-random number generator, which makes it less deterministic. The iterative core procedure, similar to the cut-down approach, rapidly eliminates reagents converging smoothly to cleaned sets while avoiding the elimination of viable reagents. We define viable reagents as those leading to only "good" products. Obtaining larger reagent sets gives more flexibility to the synthetic chemist in reagent selection, allowing them to improve the molecular diversity without sacrificing synthetic compatibility and availability, for example. The pruning procedure was also adapted to problematic cases where dramatic size differences exist between reagent sets. Excellent timings were obtained with a library of 1.5×10^{12} products that was optimized within 0.25 s on an AMD Opteron processor.[39] In the following, the data sets used to study and validate our algorithm are first presented. Next, we discuss the fast property calculation, the core algorithm, the filtering techniques, and finally the partitioning scheme.

Data Sets. The various features of our approach are investigated using the 12 different libraries illustrated in Figure 1. It was important for us to use published combinatorial libraries that have been experimentally synthesized and documented in the recent literature. For instance, Lib01,[25] Lib02,[26] Lib03,[27] Lib04,[28] Lib05,[29] Lib08,[30] and Lib10[31] are from organic synthesis types of journals. Lib03 and Lib08 can lead to different libraries that we do not consider here as explained in their respective references. On the other hand, Lib06,[32] Lib07[33] (which is an Ugi library where the fourth dimension has been fixed to a methyl), and Lib09[4] are taken from molecular modeling types of journals. Lib11 is a hypothetical tetrapeptide library designed by us in order to test the approach on very large libraries. Finally, Lib12[34] consists of a huge Ugi reaction library that extends Lib07. Diversity in initial library size, scaffolds, and functional groups were also selection criteria. The initial size of the examined libraries goes from 10^5 to 10^{12} products, as seen under the product structures in Figure 1. The reagents for all of the libraries were extracted from the ACD, using our in-house Web tool Virtual Library ToolKit,[35] to apply standard filters to eliminate metals and chemically incompatible functional groups. The number of reagents retrieved from ACD is also listed below the structures in Figure 1. Some differences in the size of primary amine, alcohol, and carboxylic acid sets in Figure 1 originate from different search criteria related to chemical compatibility. Concerning the diversity of the scaffolds, there are four cyclic and eight linear structures. The minimum number of non-hydrogen atoms goes from 2 to 13; the minimum numbers of hydrogen-bond acceptor and donor atoms (HBA and HBD) span the range of $0-4$ and $0-3$, respectively. In terms of functional group diversity, 15 different kinds of reagents were needed: alcohols, aldehydes, primary and secondary amines, sulfonyl chlorides, carboxylic acids, α-haloketones, boronic acids, aryl bromides, halides, amino acids, isocyanates, thioureas, 2-aminoethanols, and isocyanides. We believe that these libraries are representative examples of real-life problems to which the algorithm presented in this article can be applied. We apply the Lipinski rule of five[8] as a filtering rule that identifies a good product if it has less than six H-bond donors, less than 11 H-bond acceptors, less than 34

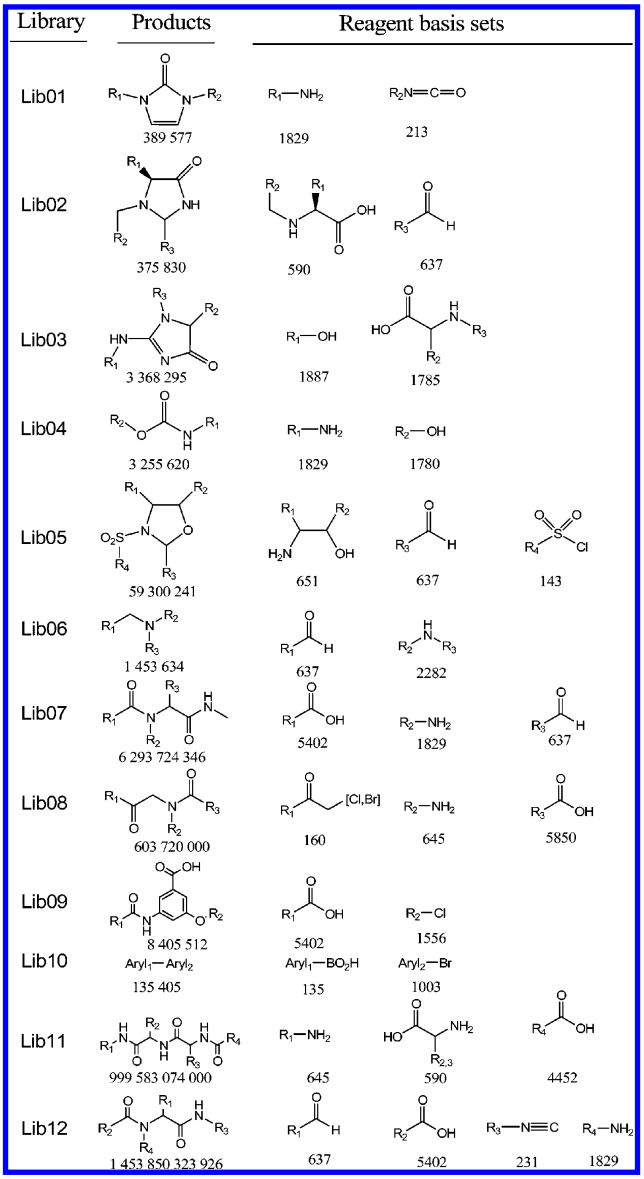

Figure 1. Twelve libraries used in this work ranging from 10^5 to 10^{12} potential products and built from two to four reagent basis sets each. Lib11 has four dimensions because the same original amino acid basis set is used in two different dimensions.

heavy atoms, and a logP between −2.4 and 5.0. The original rule of five considers that a compound with a molecular weight less than 500 Da is acceptable; by targeting 34 nonH atoms, we target a lower molecular weight, which is a way to obtain more leadlike products. An exception was made for Lib11 because of the size and nature of the template; the rule applied is the following: HBA ≤ 6, HBD ≤ 8, −2.4 ≤ logP ≤ 5.0, and nonH ≤ 34. These rules, while attempting to improve the drug-likeliness of the library, decrease the diversity of the reagents from the fact that extreme cases (i.e., large and very nonpolar reagents, large with many HBD reagents, etc.) are eliminated. Depending on the motivation for the library synthesis, other rules can be more appropriate (better starting point for lead optimization, more soluble compounds, etc.). Two-dimensional structures in MDL SD format of all of the reagents used in this work are electronically available as Supporting Information.

2. PROPERTIES CALCULATION

The most time- and memory-consuming step in any optimization workflow based on the assessment of the products is usually the calculation of the properties. Traditionally, the calculation of even simple properties such as the molecular weight, HBA, HBD, and logP require the generation of the product structures. For large combinatorial virtual libraries of a few hundred thousand products, the memory requirements to store the full library enumeration and the CPU time needed to calculate the product properties are prohibitive. There are much faster ways to obtain these properties without building the products; Shi et al.[36] have shown that the product properties can be formulated as the sum of the reagent properties plus a correction that accounts for the chemical transformation of the reagents and the scaffold. We use the simpler and more approximate of their two corrections, based on the enumeration of a single product. Because the synthetic scheme for a given library is constant, this correction should not vary too much from product to product. Although they show that the ClogP is not very well approximated (correlation coefficient of 0.902 with a single library and 9702 products) with this approach, we show that the similar Klopman and Wang logP[37] (KWlogP) property is additive within a library. The reason for this difference is not clear to us especially because they are both fragment-based approaches. The choice of KWlogP can also be rationalized by better performances compared to those with ClogP.[37] In this work, the properties are obtained using our in-house tools.[38] The property of a product is thus approximated by

$$\text{property(product)} = \sum_{d=1}^{D} \text{property(reagent}_d\text{)} + \delta\_\text{property} \quad (1)$$

where property(product) is the property of a product, property(reagent$_d$) is the property of the reagent used from the basis set (or set of reagents) of the dimension d to form that product, D is the number of dimensions in the library, and δ_property is the library-specific correction for that property. The correction can be calculated using the simple recast of eq 1:

$$\delta\_\text{property} = \text{property(product)} - \sum_{d=1}^{D} \text{property(reagent}_d\text{)} \quad (2)$$

This requires building one product structure and the calculation of its properties as well as those of the D reagents used to build it. Subsequently, calculating the property of any product of the library consists of the simple addition of eq 1 without building the chemical structure of the product. For each of the libraries 1−11, a randomly generated subset of reagents in each dimension was selected and their combinatorial products were built in order to verify the accuracy of this approximation. Lib12 is left out because it is simply an extension of Lib07. In more detail, the number of reagents selected in each basis set in a given library was chosen as follows: 32 reagents for two-dimensional libraries, 10 reagents for three-dimensional libraries, and 6 reagents for the four-dimensional library. This led to a total of 11 464 molecules for which nonH, HBA, HBD, and KWlogP

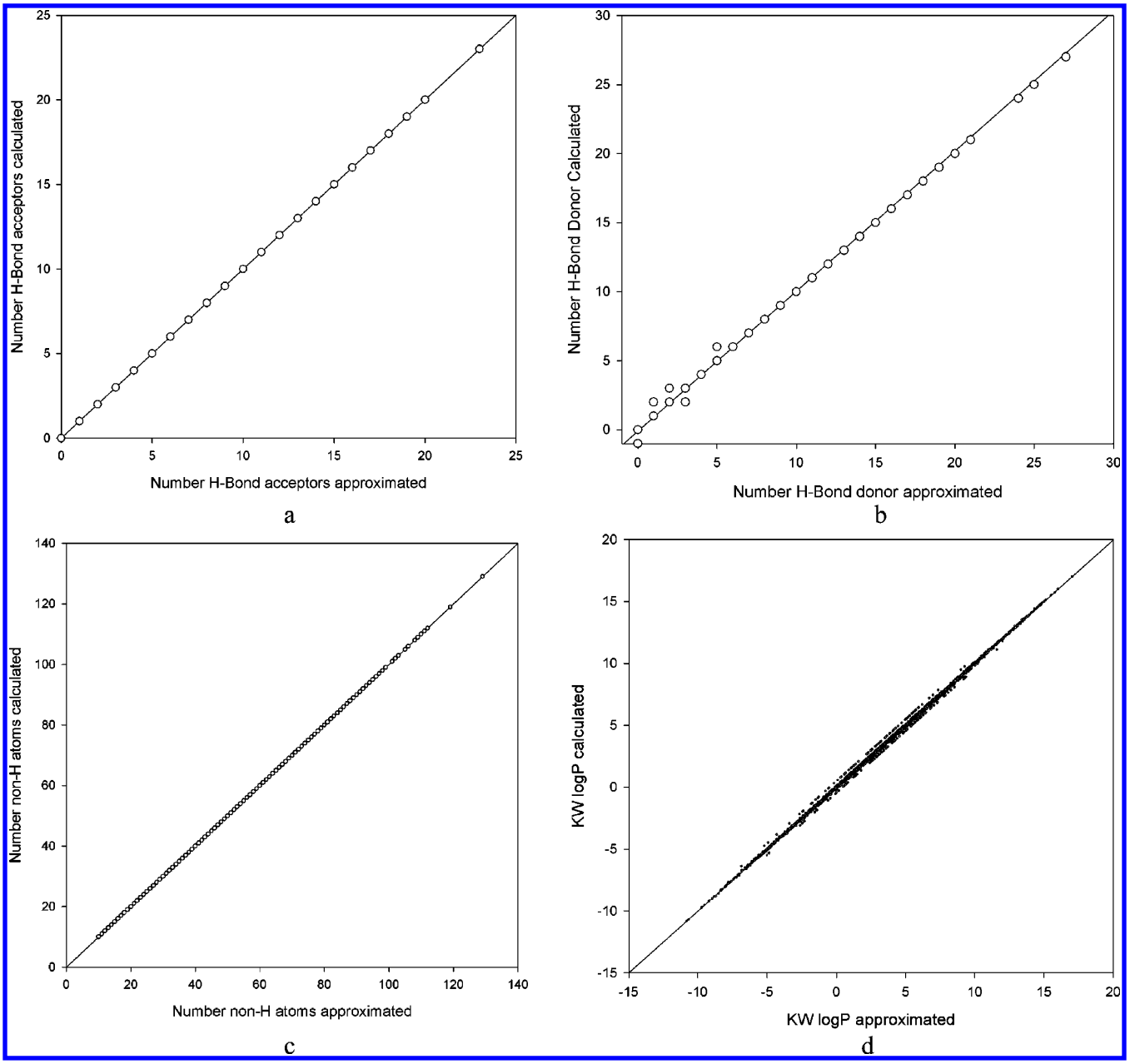

Figure 2. Four properties, (a) number of hydrogen-bond acceptors, (b) number of hydrogen-bond donors, (c) number of non-hydrogen atoms, and (d) Klopman and Wang log P, calculated using the built structures (ordinate) and the approximation from eq 1 (abscissa). Overall, 11 464 molecules from the 11 library sets are calculated.

properties were calculated. We then calculated the properties for the products and approximated them using the reagent properties with eq 1. The correction was calculated using a single product randomly selected. Figure 2 shows the correlation plot between the approximated quantity and the exact calculated counterpart. A perfect approximation would produce a line passing through the origin with a slope of 1 and a regression coefficient of 1. The four graphs show an extremely good correlation with correlation coefficients of 1, 0.996, 1, and 0.999 for HBA, HBD, nonH, and KWlogP, respectively. The additive approximation did not make mistakes for HBA or nonH and was found to be quite precise for the KWlogP estimation. The worst of the four correlations was obtained with HBD, where among 11 454 products, 632 (5.6%) were off by 1. Looking at the data in more detail, the problem lay with nitrogen-containing reagents where some but not all nitrogen atoms underwent a change in protonation state over the course of the reaction. For example, in Lib04, R_1 could include either aryl or aliphatic amines.[27] This sort of discrepancy can be anticipated when a new bond is formed with a nitrogen atom in the synthetic scheme. The hybridizations of the atoms directly connected to the reagent's nitrogen atom (i.e., sp^1, sp^2, or sp^3) should be compared with the product. With GLARE, it is possible to then avoid this problem by optimizing the whole library as if it were two sublibraries, although further details go beyond the scope of this article. The correction δ_property must be calculated using a representative example or else the error on the properties might be higher and systematic. Overall, the results in Figure 2 show that using eq 1 is an adequate approximation for high-throughput properties calculation with a minimal impact on the quality of the results given in this article. For this reason, all of the product properties are evaluated using eq 1 in this work. Shi et al.[36] did not look

into HBA and HBD additivity, but they showed that the polar surface area, solvent accessible volume, and van der Waals volume are additive as well. This shows the generality of eq 1. A tremendous advantage of this approach is that the actual calculation of the properties is performed only on the initial reagent sets prior to running GLARE and may even be stored in a database. The properties are read in during program initialization and kept in memory during the course of the execution.

3. CORE ALGORITHM

Iterative Optimization. The principal objective is to end up with a combinatorial set of virtual products that would satisfy user-defined filtering rules. Another objective is to keep the maximum number of reagents in order to leave more choices to the synthetic chemist or a subsequent filtering step. Optimally, fulfilling these two objectives would lead to a maximally effective and maximally "good" library. The effectiveness at iteration "*i*" is defined by

$$E^i = \frac{1}{D}\sum_{d=1}^{D}\frac{N_d^{\,i}}{N_d^{\,0}} \tag{3}$$

where N_d^0 is the number of reagents in the *d*th dimension of the initial set of reagents (e.g., as extracted from the ACD), N_d^i is the number of reagents in the *d*th dimension in the pruned set of reagents at iteration "*i*", and *D* corresponds to the number of dimensions. It calculates the fraction of reagents remaining relative to the number of reagents initially available. This definition of the effectiveness will be used unless otherwise mentioned in the text. This reagent-orientated metric should not be confused with Bravi et al.'s effectiveness definition,[21] which is the ratio of good products in a given iteration to total good products in the initial library. Unfortunately, the latter definition of effectiveness cannot be used in subsequent sections where the partitioning of reagent sets is considered.

The goodness is defined by

$$G^i = \frac{N^i_{\text{good}}}{N^i_{\text{products}}} \tag{4}$$

where N_{good} is the number of "good" products and N_{products} is the total number of products for a given iteration. It calculates the fraction of compounds among all the products that comply with the preset "goodness" criteria. The effectiveness as defined by eq 3 will be used to improve parameters of our algorithm with the objective of keeping the maximum number of reagents for a given goodness. A higher effectiveness does not necessarily mean more good products but, rather, that a higher proportion of the initial reagent lists is being used. In reality, maximizing goodness and effectiveness exactly is NP-hard (nondeterministic polynomial-time hard).[20]

GLARE uses an approach similar to the Stanton et al.[23] cut-down procedure. At each iteration, the monomers are scored according to their occurrence in "good" products. The worst ones are then eliminated, and the remaining ones are rescored in a subsequent iteration. The algorithm can be outlined as follows:

(i) Using eq 1, the properties of all the possible combinatorial products are calculated on the fly (the results are not kept in memory). During this process, if a product is "good" according to the filtering rules, each reagent necessary to make it has its score incremented by one.

(ii) If the "goodness" (eq 4) in the combinatorial set is above a user-defined threshold, stop; otherwise, continue to step iii.

(iii) Once all the combinatorial products are examined, rank the reagents in a given dimension according to their score.

(iv) The reagents rarely present in "good" products have a low score and are, therefore, pruned out from their individual reagent list according to a procedure outlined below. This is applied on every dimension. Each reagent score is reset to zero. Return to step i.

The library formed by the initial reagent lists is by definition maximally effective but is the least enriched in "good" products. Applying the algorithm results in a virtual library which has improved goodness but reduced effectiveness. The key question is how many reagents to reject at step iv: rejecting too many would throw out good reagents, reducing the effectiveness of the library in an unrecoverable way; rejecting too few would not sufficiently improve its goodness. Our approach was to determine an "optimal" number of reagents to reject and iterate (score the reagents, rank them, and prune the worst ones); we can observe a change in the ranking of the reagents during the process. The iterative process can then be stopped when the goodness is above a threshold noted G^t. For example, if a 95% "good" products threshold is used and the remaining monomer sets contain respectively 125, 400, and 300 reagents, then among the 15 000 000 virtual products, 14 250 000 would match the filtering rules. One could also use the remaining number of reagents to decide when enough optimization has been done. In other words, either library goodness or effectiveness can be controlled via the stop criteria. Practically, a goodness of 95% enforces that, no matter which reagents from the optimized sets are selected by a chemist expert or an algorithm, they will form a well-behaved library. The set of reagents being (often greatly) reduced, it makes the latter selection more affordable, and chemists can spend their valuable time and expertise looking at viable reagents only.

Reagent Set Pruning. The pruning algorithm in GLARE avoids the inefficiency of rejecting reagents one at a time, which would require thousands of iterations with large basis sets. The relative ranking of the reagents within a dimension does not change dramatically between iterations. Also, we noticed that many equally good reagents exist after they are scored; therefore, the choice of eliminating one in particular would be completely arbitrary as well as not leading to significant improvement. A perfect pruning strategy will keep a maximum number of filtered reagents for a given goodness. In other words, it would maximize the library effectiveness for a given goodness. In this spirit, the following equation allows the gradual elimination of reagents:

$$K^{i+1}(G^i) = \left(\frac{G^i - G^0}{G^t - G^0}\right)(1 - K^0) + K^0 \tag{5}$$

where K^{i+1} is the fraction of reagents to keep for the next iteration in all of the dimensions, K^0 is the initial fraction of

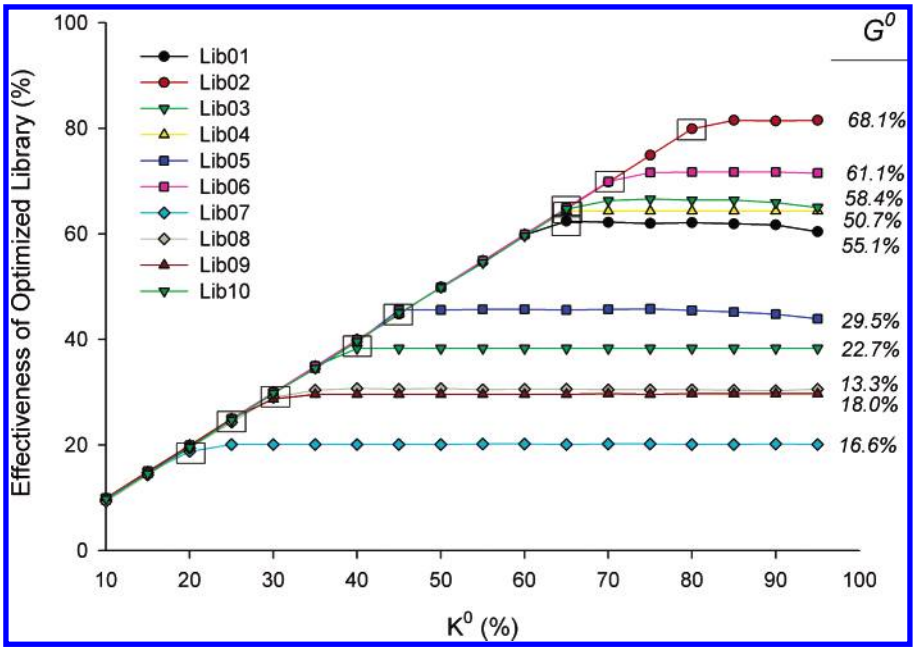

Figure 3. Final effectiveness (eq 3) of 10 libraries when optimized as a function of the initial number of reagent kept (K^0) in percent. A plateau is reached in most libraries, and the smallest K^0 part of this plateau is called the optimal K^0 and denoted $K_{opt}{}^0$; it is identified by a box for each library. Each of these boxes is identified using numerical second derivatives. The initial library goodness (G^0) is also shown for each library. This reveals that the lower the G^0, the lower the $K_{opt}{}^0$. For this particular graph, a goodness threshold (G^t) of 95% was used.

reagents to keep, G^0 is the initial goodness of the virtual library, G^i is the goodness at iteration i, and G^t is the threshold goodness of the library. The iteration index i starts with a value of zero and is incremented at each iteration. After the first round of product properties calculation within the first iteration, we set $i = 0$ such that G^i becomes G^0 and $K^1(G^0) = K^0$ for any G^0. Thus, following eq 5, a fraction K^0 of reagents in each reagent set is kept (or $1 - K^0$ is rejected) before the second iteration. At the other end of the optimization, all the reagents should be kept if the goodness reached the threshold G^t since the pruning iterations are stopped; that is, if $G^i \geq G^t$, then $K^{i+1} = 1.0$. Equation 5 is simply a linear interpolation between the two points (G^0, K^0) and (G^t, 1.0).

The outlined pruning strategy eliminates more reagents at the beginning and slows down close to the targeted goodness. By carefully choosing K^0, one can control the speed of the whole process and the final effectiveness: a value of 90% tends to maximize the effectiveness while needing a lot of iterations to reach the targeted goodness; a value of 50% reaches the desired goodness in fewer steps and less computer time, but more viable reagents might be eliminated. What is the optimal choice of K^0 for a library to minimize the number of iterations while maintaining a maximum effectiveness? Figure 3 shows the final effectiveness of the optimized libraries (Lib01 to Lib10) as a function of K^0, which is varied from 10% up to 95%. It is interesting to note that, in all cases, the effectiveness reaches a plateau. If a relationship between the optimal K^0 and G^0 can be established, then one could use the result from the first iteration to calculate the optimal K^0 ($K_{opt}{}^0$) for a particular library in an automated way. In Figure 3, small boxes show the optimal K^0 for each library, which can be automatically found using numerical second derivatives. In the graph of Figure 3, G^0 is also shown next to each library curve. By looking at these data, it is tempting to draw a relationship between $K_{opt}{}^0$ and G^0. Of course, if G^t is changed, the values

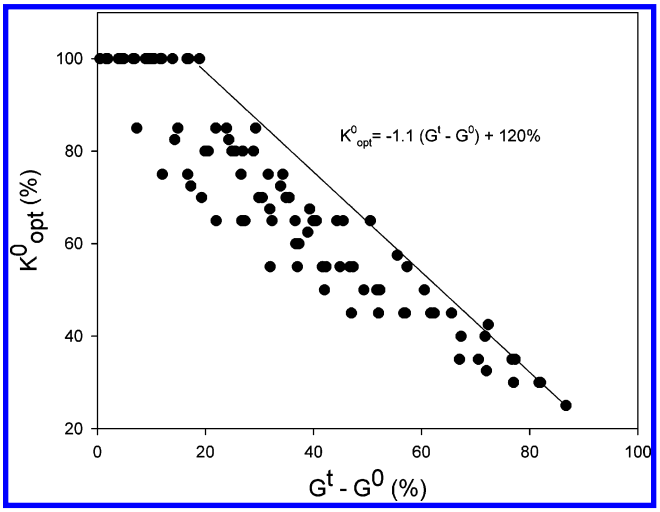

Figure 4. Optimal K^0 ($K_{opt}{}^0$) as a function of the difference between the goodness threshold and the initial goodness. Each point is derived from a curve like those of Figure 3 where the small boxes identify $K_{opt}{}^0$; the effectiveness metric used is given by eq 3. G^t takes the values 30, 40, 50, 55, 60, 65, ..., 95, 100%, and G^0 values come from Lib01 to Lib10 excluding Lib07. The linear equation shown is the upper bound of $K_{opt}{}^0$ that one should use to maximize the effectiveness of the optimized library while minimizing the computational effort.

of $K_{opt}{}^0$ would change as well; in other words, using $G^t <$ 95% would shift the plateaus of Figure 3 to higher values of K^0. Hence, we need to establish the relationship between $K_{opt}{}^0$, G^0, and G^t. Figure 4 shows $K_{opt}{}^0$ as a function of $G^t - G^0$, where this choice of the abscissa can be justified by the fact that what really matters is how far from the threshold you are. Each point on this figure comes from a curve such as those of Figure 3, and we used Lib01 to Lib10 excluding Lib07. The beginning of a plateau was identified using the numerical second derivative on all effectiveness curves. The values of G^t used were 30%, 40%, 50%, 55%, 60%, 65%,

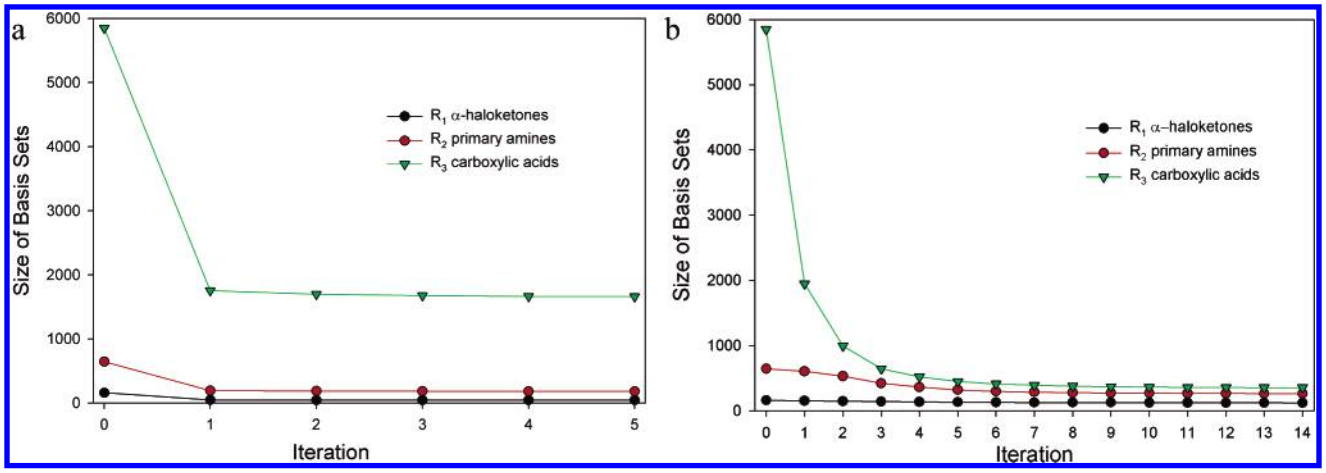

Figure 5. Scaled and nonscaled pruning for Lib08. (a) Nonscaled pruning, showing basis set sizes during the filtering process. The library is optimized within four iterations. In the original sets, 5850 acids, 645 primary amines, and 160 α-haloketones can be found. The filtered set sizes are 1661, 182, and 43, respectively. (b) The scaled pruning algorithm conserves very small basis sets at the expense of larger basis sets. From the same original sets used in part a, the filtered sets contain 355 acids, 264 amines, and 121 α-haloketone reagents, respectively. In both cases, a goodness threshold of 95% and $K^0 = K_{opt}{}^0$ ($G^t = 95\%$ and $G^0 = 13.3\%$) = 30.1% are used; in part b, the logistic function exponent α is set to 6.0.

..., 100%, and the G^0 came from the libraries' initial goodnesses. Figure 4 shows a wide distribution of points, and we choose the upper bound to obtain the linear equation $K_{opt}{}^0(G^t,G^0) = -1.1(G^t - G^0) + 120\%$; if $G^t - G^0 \leq 23\%$, $K_{opt}{}^0$ is set to 95%. This equation should be of general use and independent of the filtering rules. We also wanted to check the dependency of this equation on the effectiveness definition. For this purpose, we obtained the relationship using Bravi's product-based effectiveness:[21] $K_{opt}{}^0(G^t,G^0) = -0.9(G^t - G^0) + 108\%$, which leads to systematically smaller $K_{opt}{}^0$ values. For the remainder of this article, the first equation is used.

Scaled Reagent Set Pruning. The above pruning approach considers all the dimensions equally. We have found that it is undesirable for a large set of reagents (e.g., amines) to be pruned at the same rate as a much smaller set (e.g., boronic acids). In such a situation, the smaller set becomes so small that one is not able to select enough reagents for the synthesis. In addition, the initial poor structural diversity of many of the small reagent sets is amplified when too few of them remain in the resulting pruned sets. For this reason, we designed a scaled pruning technique that initially prunes larger sets faster. The following equation turns on the K^i of the largest dimension first, and as it gets closer to the size of the next largest list, it turns on its K^i:

$$K_d^{i+1} = 1 - \frac{1 - K^{i+1}(G^i)}{1 + \exp[-\alpha(S_d^i/\max_l S_l^i - 0.5)]} \qquad (6)$$

where K_d^{i+1} is the fraction of reagents to keep in the basis set of dimension d, $K^{i+1}(G^i)$ is given by eq 5, α is related to the steepness of the slope in the switchover portion of the logistic function, and S_d^i is the size of the dth basis set at iteration i. We found that setting α to 6.0 has the effect of sufficiently enforcing the reduction of the largest dimension first while leading to good effectiveness. The advantage of using eq 6 instead of eq 5 only is shown in Figure 5 where the number of remaining monomers is plotted at each iteration using eqs 5 and 6 for Lib08. The reagent sets for this library are of dramatically different sizes: 160 α-haloketones, 645 primary amines, and 5850 acids (cf. Figure 1). The graph of Figure 5a, where eq 5 is applied, shows that only 43 α-haloketones remain when a goodness of 95% is reached. We find these molecular structures not diverse enough to allow the selection of 10 or 20 monomers. On the other hand, in Figure 5b, the scaled pruning using eq 6 is much more forgiving toward the α-haloketones, leaving 121 reagents at the expense of the large carboxylic acid set, which is reduced from 1661 to 355 reagents. Although the larger sets get smaller with the scaled pruning, they are still likely to contain enough compounds to make the final selections. The scaled pruning algorithm tends to equalize monomer set sizes. This has been very useful in some of our in-house library design cases. This way of biasing the pruning can also be used to enforce other restraints, for example, to increase the number of viable reagents in the dimension from which a larger amount of reagents is needed for synthesis. In cases where a large difference between the reagent list sizes exists, the scaled pruning usually increases the effectiveness of the optimized library simply because it affects mainly one term in eq 3.

4. PARTITIONING TECHNIQUE

The fast property calculation, the iterative algorithm, and the pruning scheme are rapid and efficient, but large virtual combinatorial libraries were still intractable. For example, Lib07, Lib11, and Lib12 would lead to $\sim 10^9$ and $\sim 10^{12}$ possible products. Beroza et al.[24] have utilized a random sampling technique to approximate the goodness of a monomer on a per dimension basis. With random sampling, they could filter a 13 billion product Ugi library. Here, a different approach is proposed where all the reagents from all dimensions are scored at one time through the partitioning of the basis sets. Instead of performing a full combination of the reagents, we do a partial combination using partitions of the reagent basis sets from each dimension and attribute scores to the monomers.

Partitioning. For example, given a three-dimensional virtual library with three reagent lists A, B, and C, the objective is to score and rank the reagents $a_i \in$ A in order to

identify and eliminate the worst ones (similarly for $b_i \in B$ and $c_i \in C$). Instead of scoring the entire $N_A \times N_B \times N_C$ product matrix (where N_A is the number of reagents in set A), we can partition the sets A, B, and C into p_A, p_B, and p_C subsets, respectively. Then, it is possible to calculate the properties of much smaller subsets at a time, corresponding to smaller blocks of the product matrix. For instance, if $p_A = p_B = p_C = 2$, then these two combinatorial sublibraries could be considered $A_1 \times B_1 \times C_1 = P_1$ and $A_2 \times B_2 \times C_2 = P_2$ where A_i is a partition subset of the reagent set A and P_i is the set of products formed by the ith combination. It is a much smaller burden computationally to score the reagents on the basis of the number of "good" products they form in their respective P_i sets. Assuming the sets are partitioned equally, 4-fold fewer property evaluations are needed compared to the full combination. Once the reagents are scored within their partition, then the scores can be compared directly between partitions. The idea behind this approach is based on the hypothesis that each subset P_i should have approximately the same property distribution as the whole P set, corresponding to a statistical sample of the combinatorial library. Then, the reagent ranking obtained with the partition-based combination would be very similar to the one obtained with the full reagent combination. This partitioning technique can reduce considerably the number of property calculations. In practice, we used a partitioning scheme that enforces all product subsets formed by the partitions to have equal or close to equal sizes, as explained in more detail in the Appendix. This reagent scoring strategy is incorporated in the iterative procedure. After the reagent lists have been pruned, the reagents are randomly permuted before the next iteration. The G^i used in the K^i calculation is approximated using the fraction of good products encountered during the process of combining sublibraries; in eq 4, $N_{product}$ becomes the number of evaluated products.

Figure 6 shows an overview of the GLARE workflow when the partitioning scheme is used. Few elements need to be provided in the input: the reagent properties grouped by dimension and the property corrections. The reagent properties are precalculated once for all reagents in ACD, and one correction per property per library needs to be calculated. The default parameters are $R_{min} = 16$ for the partition sizes, pruning (not scaled) with K_{opt}^0, and $G^t = 95\%$. If scaled pruning is used, the default α is set to 6.0. The optimization is performed as explained above with some extra steps related to the partitioning scheme. The first step is to create many sublibraries that are subsequently used to score each reagent according to their presence in "good" products within the sublibraries. If the goodness is below the user-defined threshold, all the reagents are put back in their original lists and ranked. Equation 5 or 6 is used to identify the reagents to eliminate, and the new lists are shuffled in order to avoid reforming similar partitions. All reagent scores are reset to zero, and iterations continue until the goodness threshold is reached.

The question remains: how small can the partitions be, and how does it affect the results? The maximum speedup is obtained with the smallest partition size (i.e., the largest number of partitions), but when the partition size is too small, the reagent score becomes more approximate; thus, there is a tradeoff between speed and score quality. Figure 7 addresses this question. An optimization with partitioning

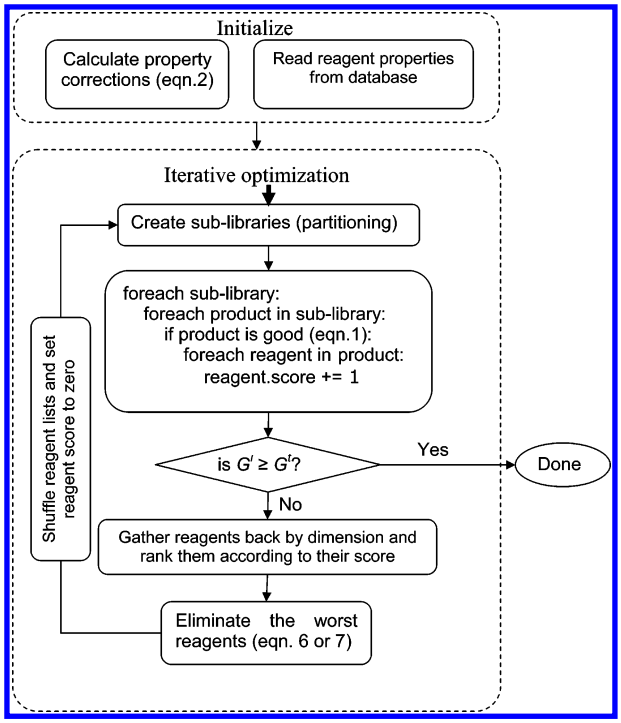

Figure 6. Algorithm summary for the whole optimization process when partitioning is used. The blocks correspond to logical tasks, and the arrows show execution precedence. The users need to define parameters: minimum partition sizes, K^0 (K_{opt}^0 can be used as a default), whether to use the scaled pruning together with the α parameter, and the goodness threshold G^t that can differ from 95%.

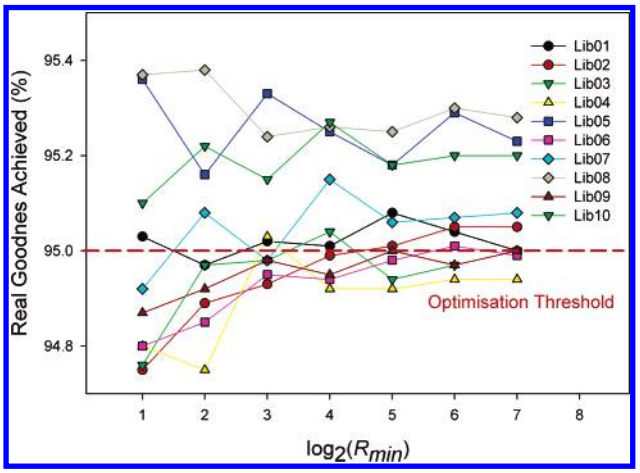

Figure 7. Optimization first performed using the partitioning scheme with a target of 95% goodness and K_{opt}^0. Using the resulting reagents, the percent of good products is evaluated in consideration of all the products. This figure shows the percent of "real" good products obtained with several partition sizes. Ten randomly different runs are used to calculate the shown average values. Including the standard deviations (not reported), the lowest data point is at 94.4% and the highest is at 96%.

was performed on all libraries using a target goodness of 95%. K^0 was calculated with the previously established formula (cf. Figure 4). The minimum partition size (R_{min}) is shown on the abscissa in logarithmic scale ranging from 2 to 128 reagents per partition. After the partitioned optimization was performed, we calculated the true goodness of the resulting library to examine differences from the approximation obtained in the partitioning regime. A total of 10 different initial partition arrangements were randomly produced, and statistics were collected through 10 separate runs.

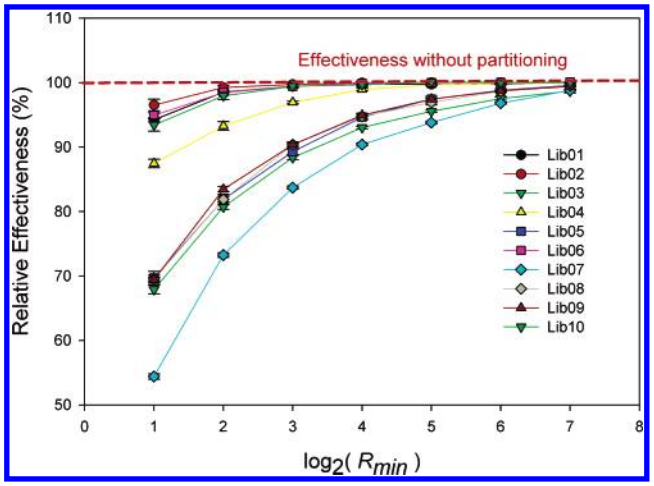

Figure 8. Assessment of the effect of the partitioning scheme on the library effectiveness. The effectiveness of the library is optimized with the partitioning scheme normalized to the effectiveness of the library pruned using the full combination and plotted for several R_{min}'s. Ten randomly different trials were done, and the standard deviation is reported with error bars. Overall, the smaller the partitions, the fewer the reagents left in the optimized library.

The ordinate axis of Figure 7 shows the goodness of the fully combined optimized library. The partitioning scheme delivers a true goodness very close to 95% for all of the libraries and for all of the partition sizes: the goodness is conserved upon partitioning. These runs are independent and allow us to characterize the variability of the outcome. Of course, we do not need to do more than one run to optimize a library in practice.

How is the effectiveness of the libraries conserved upon partitioning? To answer this question, the effectiveness of the library pruned with the partitioning scheme is normalized to the effectiveness of the library pruned without partitioning and is plotted for several R_{min} in Figure 8. Statistics were also accumulated with 10 initial starting points. In this figure, K_{opt}^0 is utilized as is a goodness threshold of 95%. It is clear that partitioning impacts the effectiveness of the result. In fact, in the worst case of Lib07, the size of the resulting reagent sets is only 54% of the one obtained with a full combination. This figure shows also that, with R_{min} equal to or higher than 16, the resulting sets obtained with the partitioning scheme contain 90% as many reagents as the sets from the full combination. Therefore, R_{min} should be set to 16 or higher if the user needs to maximize the number of remaining reagents. An interesting alternative is to use the extremely fast $R_{min} = 2$ or 4 while setting $K^0 = 80\%$ or higher. The resulting effectiveness is found to be reasonably close to the best possible effectiveness obtained without partitioning while being invariably faster. Overall, partitioning does lead to good libraries but tends to reduce the effectiveness compared to the nonpartitioned case.

Computational Results. All of the calculations were carried out on an IBM Linux workstation.[39] GLARE was written in C++ and compiled using a gcc 3.3.5 compiler from SUSE Linux with the full optimization option −O4. Table 1 shows the timings for the algorithm with and without the partitioning scheme. The 12 libraries were considered, and the reagent pruning was run with K_{opt}^0 as determined with the equation of Figure 4, and the goodness threshold was set to 95%. The number of necessary properties/product evaluations is shown for the partitioning scheme as well. It is clear that even immense libraries such as Lib12 can be optimized within a reasonable time frame, achieving dramatic speedups with the partitioning algorithm. The timings were averaged over 10 separate runs, and we report the 95% confidence probability error on the basis of the *t*-test. Most of the calculations were done within 1 s, and the toughest calculation (Lib12) was performed within 1 min 25 s. The latter result is quite impressive considering the initial size of this library (>1.4 trillion products). With this particular library, reducing R_{min} to 8 decreases the calculation time by a factor of 10 down to 10 s; when $R_{min} = 2$, the required time goes below 1 s! In a practical application, it would be wise to precalculate the number of evaluations needed and adjust R_{min} accordingly. As shown in Figures 7 and 8, the goodness of the resulting library is maintained at 95%, because the timings were obtained from the same runs. These remarkable timings make GLARE a good algorithm to use where a fast response is an important requirement, such as in a Web-based interface.

Table 1. Timings in Seconds for Optimizing the 12 Libraries Using the Standard Full Combination Technique and the Partitioning Scheme with $R_{min} = 16$[a]

library	size	partitioning (s)[b]	number evaluations in partitioning[b]	full combination (s)[b]
Lib01	389 577	0.11 ± 0.03[c]	86 898	2.11 ± 0.01[c]
Lib02	375 830	0.05 ± 0.01	18 568	3.29 ± 0.02
Lib03	3 368 295	0.110 ± 0.008	63 683	6.92 ± 0.04
Lib04	3 255 620	0.11 ± 0.02	70 977	16.3 ± 0.2
Lib05	59 300 241	0.51 ± 0.07	452 481	89 ± 1
Lib06	1 453 634	0.12 ± 0.02	87 024	11.07 ± 0.08
Lib07	6 293 724 346	8.1 ± 0.2	5 343 428	6394.4 ± 0.9
Lib08	603 720 000	4.3 ± 0.3	4 020 577	621.2 ± 0.7
Lib09	8 405 512	0.23 ± 0.03	158 583	15.8 ± 0.2
Lib10	135 405	0.110 ± 0.006	38 852	0.66 ± 0.02
Lib11	999 583 074 000	26.3 ± 0.6	31 244 103	>12 days[d]
Lib12	1 453 850 323 926	85 ± 4	72 328 208	>17 days[d]
	$R_{min} = 8$	9.92 ± 0.2	7 767 614	
	$R_{min} = 4$	1.30 ± 0.2	959 861	
	$R_{min} = 2$	0.245 ± 0.009	110 824	

[a] K_{opt}^0 is utilized with the threshold goodness set to 95%. The reported timings were obtained on an IBM Linux workstation.[39] [b] Averaged over 10 runs; random initial conditions were used when partitions were needed. [c] Error estimates were obtained from a t-test using a 95% confidence probability. [d] Approximated from Lib07's computation time per product evaluation (1.03 μs), considering one iteration.

Table 2. Reagent List Sizes of the Initial Libraries (a) Compared to the Optimized Libraries with $R_{min} = 16$ without Scaled Pruning (b) and with Scaled Pruning Where α Is Set to 6.0 (c)[a]

library	R_1		R_2		R_3		R_4	products	E (%)	G^0 (%)
					Lib01					
a	2056	×	213					437 928	100.0	55.1
b	1291	×	130					167 830	61.9	
c	1063	×	192					204 096	70.9	
					Lib02					
a	590	×	637					375 830	100.0	68.1
b	479	×	517					247 643	81.2	
c	479	×	515					246 685	81.0	
					Lib03					
a	1887	×	1785					3 368 295	100.0	22.7
b	672	×	635					426 720	35.6	
c	675	×	653					440 775	36.2	
					Lib04					
a	1829	×	1780					3 368 295	100.0	50.7
b	1163	×	1133					1 317 679	63.6	
c	1161	×	1136					1 318 896	63.6	
					Lib05					
a	647	×	637	×	143			58 935 877	100.0	29.5
b	284	×	279	×	60			4 754 160	43.2	
c	226	×	223	×	109			5 493 382	48.7	
					Lib06					
a	637	×	2282					1 453 634	100.0	61.1
b	452	×	1634					738 568	71.3	
c	535	×	1291					690 685	70.3	
					Lib07					
a	5402	×	1829	×	637			6 293 724 346	100.0	16.6
b	951	×	320	×	109			33 170 880	17.4	
c	378	×	387	×	323			47 250 378	26.3	
					Lib08					
a	160	×	645	×	5850			603 720 000	100.0	13.3
b	43	×	182	×	1661			12 998 986	27.8	
c	121	×	264	×	355			11 340 120	40.9	
					Lib09					
a	5402	×	1556					8 405 512	100.0	18.0
b	1532	×	440					674 080	28.3	
c	962	×	743					714 766	32.8	
					Lib10					
a	135	×	1003					135 405	100.0	58.4
b	88	×	667					58 696	65.8	
c	116	×	511					59 276	68.4	
					Lib11					
a	645	×	590	×	590	×	4452	999 583 074 000	100.0	0.60
b	84	×	77	×	77	×	598	297 825 528	13.1	
c	120	×	124	×	124	×	106	195 582 720	15.8	
					Lib12					
a	637	×	5402	×	231	×	1829	1 453 850 323 926	100.0	0.32
b	57	×	492	×	19	×	166	88 450 776	8.8	
c	110	×	102	×	98	×	101	111 055 560	16.8	

[a] In all optimized cases, the resulting goodness is 95% ± 1%.

It is possible to compare these results with the work of Beroza et al.[24] and their Lockdown approach. They have optimized a 9 × 461 × 2285 × 1372 Ugi library and needed only 185 000 property evaluations of products. In their case, the first dimension was kept frozen; therefore, Lib07 seems to be the closest comparator from this work. Using GLARE, a goodness of 100% was obtained with 95 672 property evaluations in 0.019 s ($R_{min} = 2$, $K^0 = K_{opt}{}^0$, and $G^t = 100\%$). The resulting library had an effectiveness of 5.9% compared to 1.4% for Lockdown, as shown in Table 1 of their work, although their library had a pass rate (goodness) close to 30% initially compared to 5.26% for Lib07. Therefore, in this specific comparison, GLARE shows advantages regarding the number of product evaluations needed and the number of reagents kept. GLARE has also shown very interesting performance with Lib12 because 110 824 evaluations on average were needed to achieve a goodness of 95% in 0.25 s.

Finally, Table 2 shows the size of the reagent lists obtained before (a) and after (b) applying GLARE with the minimum partition size set to 16. The same calculations were repeated with scaled pruning with α set to 6.0. For all of these library optimizations, a goodness threshold of 95% was used together with $K_{opt}{}^0 = -1.1(G^t - G^0) + 120\%$. These are the optimum parameters for the algorithm presented in this article. From Table 2, we can appreciate how GLARE

reduces the size of the initial reagent lists. Although reagent-based filtering rules could accomplish the same size reduction task, it becomes clear that the product-based filtering brings a useful specificity for scaffolds and dimensionality. For example, the aldehydes present in Lib02, Lib05, Lib06, Lib07, and Lib12 contain initially 637 reagents, which turn out to give 517, 279, 452, 109, and 166, respectively, in the optimized sets. In contrast to reagent-based filtering, the products formed by any selected reagent will fulfill the desired criterion. The scaled pruning also plays an important role in that it avoids overly depleting certain dimensions, for example, in Lib05, Lib08, and Lib12. This is important for the subsequent reagent selection of the library design because too few reagents would not leave enough choice, whereas in large lists, too many reagents could not be efficiently considered. In fact, combinatorial chemists reported to us that considering lists larger than 600 monomers is difficult. Moreover, product-based algorithms that calculate more computationally demanding properties than those considered in this article are likely to perform poorly on large sets. Therefore, the scaled pruning helps address both issues at once. Last, scaled pruning almost always increase the effectiveness.

CONCLUSION

GLARE is a robust and general algorithm to optimize reagent lists in the design of combinatorial libraries for matrix synthesis. It is fast enough so that large reagent lists straight out from ACD queries can be treated directly. The resulting cleaned reagent lists, often greatly reduced, are highly enriched in compounds leading to products satisfying pre-established "goodness" criteria such as the Lipinski rule of five. The selected candidates for synthesis from the cleaned lists will, therefore, necessarily produce a library with a good profile.

Fast property evaluations are the principal factor in the impressive timings for GLARE, taking advantage of the fact that many product properties in a combinatorial library can be treated as the sum of the reagent property plus an overall correction for the scaffold and connection to it. In this work, we have shown that it is true for HBA, HBD, nonH, and KWlogP, but the same strategy could be applied to evaluate product prices, polar surface area, and other useful properties. In this work, we did not address the important issue of molecular diversity; this can be treated with more time-intensive computer programs as a second step after GLARE has eliminated the reagents that could not be part of a "good" library.

The optimization itself works through a greedy iterative algorithm that was shown to converge to good solutions. The idea behind it is to score each reagent on the basis of its likelihood to lead to "good" products. The worst reagents are then eliminated before the next iteration. We have found that a goodness of 95% is largely sufficient by keeping just a few reagents on the border of what is considered as viable. However, it is possible to reduce the goodness threshold with the aim of obtaining more reagents. A different stop criterion such as a minimum threshold for the effectiveness could be used as well.

One of the main issues that we addressed concerns the effectiveness of the pruned library, meaning what proportion of the initial reagent list we keep. We found that it is important to progressively eliminate fewer reagents as the calculation evolves toward the final solution. We could establish an equation relating the optimal number of reagents to, at first, keep iteration to the initial goodness of the library and the goodness threshold. Using this equation minimizes the computational time while maintaining an optimum effectiveness. However, this pruning strategy overly penalized small reagent sets when reagent lists of large and small sizes where optimized together. This problem was dealt with using a scaled pruning scheme that eliminates reagents proportional to the list size. This was found to be extremely valuable in cases such as Lib08, which has a large disparity in the size of the initial reagent lists. Biasing the pruning can be useful in a variety of other situations such as when many more reagents need to be selected from one specific dimension.

Partitioned sampling is also an essential feature for the speedup of GLARE. It is based on the evaluation of sublibraries obtained by partitioning the reagent lists. In contrast to the random sampling method of Beroza et al.,[24] partitioning ensures that all remaining reagents will be examined at each iteration, while performing an even lower number of property evaluations. We found that, while small partitions reduced the computational burden and still gave good libraries, the overall effectiveness was reduced. To balance this, a minimum partition size of 16 reagents is recommended. An efficient alternative is to use a minimum partition size of two and set K^0 to 80% or higher.

The performances achieved in this work are better than those previously obtained by other groups[23,24] with similar algorithms; for example, GLARE can optimize large four-dimensional libraries using multiobjective rules within one second on affordable computers. We propose this method for general use in the fast and effective filtering of reagent sets to conform to a given set of product properties.

ACKNOWLEDGMENT

The authors would like to thank Bradley Feuston, Robert Nachbar, and Carl Berthelette for their invaluable comments on the manuscript. We are also grateful to MDL, who allowed us to publish in the Supporting Information the molecular structures of all of the reagents used in this work.

APPENDIX

How Are the Partitions Formed and Combined? Let us say that we need to partition a D dimension library containing $N_1, N_2, ..., N_D$ reagents where $N_1 \leq N_2 \leq N_3 \leq ... \leq N_D$. In addition, we define R_{min} as the minimum number of reagents per partition which sets a lower bound for the size of a partition. The number of partitions is calculated as follows:

$$p_1 = \left\lfloor \frac{N_1}{R_{\min}} \right\rfloor \tag{7}$$

$$p_i = \left\lfloor \frac{N_i}{R_{\min}} \right\rfloor - \left\lfloor \frac{N_i}{R_{\min}} \right\rfloor \bmod p_{i-1} \tag{8}$$

$$p_D = p_{D-1} \quad (9)$$

where the special brackets mean the "largest integer smaller than" (or floor function), "mod" stands for the modulo function, which returns the remainder after the division, and p_i is the number of partitions in dimension i. By construction, the number of partitions in a given dimension is a factor of the number of partitions found in all of the smaller reagent lists. The largest reagent list needs to be split into as many partitions as the second largest list for reasons that will be explained below. The combination of the partitions from each dimension is then fairly simple and is illustrated using a four-dimensional library where $p_1 = 3$, $p_2 = 6$, $p_3 = 12$, and $p_4 = 12$. Table 3 shows how sublibraries would be formed through combining the partitions of reagent lists. The "sublibraries" column shows the tuple (i, j, k, l), meaning partition i of the first dimension is combined with partitions j, k, and l from the other dimensions to form a sublibrary. Each other column of the table enumerates the indexes i, j, k, and l of the partitions obtained in splitting a reagent list. Notice that a reagent partition can be used more than once. From Table 3, it is easy to understand that, if the fourth dimension was split in 24 instead of 12, a factor of 2 would be gained from the reduced size of the sublibraries. However, twice as many sublibraries would be needed, and the net speedup would be null, hence, the decision to set $p_4 = p_3$.

Table 3. How the Sublibraries Are Formed When $p_1 = 3$, $p_2 = 6$, $p_3 = 12$, and $p_4 = 12$[a]

partition indices in dimension				
R_1	R_2	R_3	R_4	sublibraries
1	1	1	1	(1,1,1,1)
2	2	2	2	(2,2,2,2)
3	3	3	3	(3,3,3,3)
1	4	4	4	(1,4,4,4)
2	5	5	5	(2,5,5,5)
3	6	6	6	(3,6,6,6)
1	1	7	7	(1,1,7,7)
2	2	8	8	(2,2,8,8)
3	3	9	9	(3,3,9,9)
1	4	10	10	(1,4,10,10)
2	5	11	11	(2,5,11,11)
3	6	12	12	(3,6,12,12)

[a] Each column represents a dimension R where the partition indexes are listed; the right-most column shows the corresponding sublibraries formed by combining the partitions from the row.

Supporting Information Available: Molecular structures of all of the reagents used in this work are in MDL format and provided in a ZIP file. This information is available free of charge via the Internet at http://pubs.acs.org. Structures extracted with permission of Elsevier MDL from MDL Available Chemicals Directory (MDL ACD); see http://www.mdl-.com/products/experiment/available_chem_dir for further information.

REFERENCES AND NOTES

(1) Leach, A. R.; Hann, M. M. The in silico World of Virtual Libraries. *Drug Discovery Today* **2000**, *5*, 326−336.

(2) Boyd, D. B.; Agrafiotis, D. K.; Martin E. J. Introduction and Foreward to the Special Issue on Combinatorial Library Design. *J. Mol. Graphics Modell.* **2000**, *18*, 317−319.

(3) *Available Chemicals Directory (ACD)*; Molecular Design Limited: San Leandro, CA.

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

(5) Jamois, E. A.; Hassan, M.; Waldman, M. Evaluation of Reagent-based and Product-based Strategies in the Design of Combinatorial Library Subsets. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 63−70.

(6) Waldman, M.; Li, H.; Hassan, M. Novel Algorithms for the Optimization of Molecular Diversity of Combinatorial Libraries. *J. Mol. Graphics Modell.* **2000**, *18*, 412−426.

(7) Gillet, V. J. Reactant- and Product-based Approaches to the Design of Combinatorial Libraries. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 371−380.

(8) 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.

(9) Gillet, V. J.; Willet, P.; Bradshaw, J. Selecting Combinatorial Libraries to Optimize Diversity and Physical Properties. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 169−177.

(10) Gillet, V. J.; Khatib, W.; Willet, P.; Fleming, P. J.; Green, D. V. S. Combinatorial Library Design Using a Multiobjective Genetic Algorithm. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 375−385.

(11) Brown, R. D.; Martin, Y. C. Designing Combinatorial Library Mixtures Using a Genetic Algorithm. *J. Med. Chem.* **1997**, *40*, 2304−2313.

(12) Pozzan, A.; Leach, A.; Feriani, A.; Hann, M. Virtual Optimization of Chemical Libraries Using Genetic Algorithm. *Abstracts of Papers*; 218th National Meeting of the American Chemical Society, New Orleans, LA, Aug. 22−26, 1999.

(13) Micheli, F.; Degiorgis, F.; Feriani, A.; Paio, A.; Pozzan, A.; Zarantonello, P.; Seneci, P. A. Combinatorial Approach to [1,5]Benzothiazepine Derivatives as Potential Antibacterial Agents. *J. Comb. Chem.* **2001**, *3*, 224−228.

(14) Sheridan, R. P.; SanFeliciano, S. G.; Kearsley, S. K. Designing Targeted Libraries with Genetic Algorithms. *J. Mol. Graphics Modell.* **2000**, *18*, 320−334.

(15) Reynolds, C. H.; Tropsha, A.; Pfahler, L. B.; Druker, R.; Chakravorty, S.; Ethiraj, G.; Zheng, W. Diversity and Coverage of Structural Sublibraries Selected Using the SAGE and SCA Algorithms. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1470−1477.

(16) Zheng, W.; Hung, S. T.; Saunders, J. T.; Seibel, G. L. Piccolo: A Tool for Combinatorial Library Design via Multicriterion Optimization. *Pac. Symp. Biocomput.* **2000**, *5*, 585−596.

(17) McKenna, J. M.; Halley, F.; Souness, J. E.; McLay, I. M.; Pickett, S. D.; Collis, A. J. An Algorithm-Directed Two-Component Library Synthesized via Solid-Phase Methodology Yielding Potent and Orally Bioavailable p38 MAP Kinase. *J. Med. Chem.* **2002**, *45*, 2173−2184.

(18) Brown, R. D.; Hassan, M.; Waldman, M. Combinatorial Library Design for Diversity, Cost Efficiency, and Drug-like Character. *J. Mol. Graphics Modell.* **2000**, *18*, 427−437.

(19) Le Bailly de Tilleghem, C.; Beck, B.; Boulanger, B.; Govaerts, B. A Fast Exchange Algorithm for Designing Focused Libraries in Lead Optimization. *J. Chem. Inf. Model.* **2005**, *45*, 758−767.

(20) Wood, D. R. An Algorithm for Finding a Maximum Clique in a Graph. *Oper. Res. Lett.* **1997**, *21*, 211−217.

(21) Bravi, G.; Green, D. V. S.; Hann, M. M.; Leach, A. R. PLUMS: A Program for the Rapid Optimization of Focused Libraries. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 1441−1448.

(22) Zheng, W.; 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*, 251−258.

(23) 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.

(24) Beroza, P.; Bradley, E. K.; Eksterowicz, J. E.; Feinstein, R.; Greene, J.; Grootenhuis, P. D. J.; Henne, R. M.; Mount, J.; Shirley, W. A.; Smellie, A.; Stanton, R. V.; Spellmeyer, D. C. Applications of Random Sampling to Virtual Screening of Combinatorial Libraries. *J. Mol. Graphics Modell.* **2000**, *18*, 335−342.

(25) Cheng, J.-F.; Kaiho, C.; Chen, Mi; Arrhenius, T.; Nadzan, A. A Traceless Solid-Phase Synthesis of 2-Imidazolones. *Tetrahedron Lett.* **2002**, *43*, 4571−4573.

(26) Rinnová, M.; Vidal, A.; Nefzi, A.; Houghten, R. A. Solid-Phase Synthesis of 1,2,5-Trisubstituted 4-Imidazolidinones. *J. Comb. Chem.* **2002**, *4*, 209−213.

(27) Yang, K.; Lou, B.; Saneii, H. Rapid Assembly of 2-Aminoimidazolones on Solid Support. *Tetrahedron Lett.* **2002**, *43*, 4463−4466.

(28) Fernandez-Forner, D.; Huerta, J. M.; Ferrer, M.; Casals, G.; Ryder, H.; Giralt, E.; Albericio, F. Solid-Phase Syntheses of N-substituted Carbamates. Reaction Monitoring by Gel-Phase ^{13}C NMR Using a ^{13}C Enriched BAL-Linker. *Tetrahedron Lett.* **2002**, *43*, 3543−3546.

(29) Conde-Frieboes, K.; Schjeltved, R.; Breinholt, J. Diastereoselective Synthesis of 2-Aminoalkyl-3-sulfonyl-1,3-oxazolidines on Solid Support. *J. Org. Chem.* **2002**, *67*, 8952−8957.

(30) Cobb, J. M.; Grimster, N.; Khan, N.; Lai, J. Y. Q.; Payne, H. J.; Payne, L. J.; Raynham, T.; Taylor, J. Parallel Synthesis of 1,2,4-Trisubstituted

Imidazoles via N-Alkyl-N-(β-keto)amides Using a Carbazate Linker. *Tetrahedron Lett.* **2002**, *43*, 7557−7560.
(31) Hebel, A.; Haag, R. Polyglycerol as a High-Loading Support for Boronic Acids with Application in Solution-Phase Suzuki Cross-Coupling. *J. Org. Chem.* **2002**, *67*, 9452−9455.
(32) Agrafiotis, D. K. A Constant Time Algorithm for Estimating the Diversity of Large Chemical Libraries. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 159−167.
(33) Jamois, E. A.; Lin, C. T.; Waldman, M. Design of Focused and Restrained Subsets from Extremely Large Virtual Libraries. *J. Mol. Graphics Modell.* **2003**, *22*, 141−149.
(34) Ugi, I.; Werner, B.; Dömling, A. The Chemistry of Isocyanides, Their Multicomponent Reactions and Their Libraries. *Molecules* **2003**, *8*, 53−66.
(35) Feuston, B. P.; Chakravorty, S. J.; Conway, J. F.; Culberson, J. C.; Forbes, J.; Kraker, B.; Lennon, P. A.; Lindsley, C.; McGaughey, G. B.; Mosley, R.; Sheridan, R. P.; Valenciano, M.; Kearsley, S. K. Web Enabling Technology for the Design, Enumeration, Optimization and Tracking of Compound Libraries. *Curr. Topics Med. Chem.* **2005**, *5*, 773−783.
(36) Shi, S.; Peng, Z.; Kostrowicki, J.; Paderes, G.; Kuki, A. Efficient Combinatorial Filtering for Desired Molecular Properties of Reaction Products. *J. Mol. Graphics Modell.* **2000**, *18*, 478−496.
(37) Klopman, G.; Li, J.-Y.; Wang, S.; Dimayuga, M. Computer Automated log P Calculations Based on an Extended Group Contribution Approach. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 752−781.
(38) Personal communication with Robert Sheridan.
(39) The workstation used is an IBM Intellistation Pro including dual AMD Opteron 248 processors with a clock speed of 2.6 GHz and 4 GB of PC3200 ECC memory. The board's core logic chipset is an AMD 8000 series with Direct Connect Architecture and HyperTransport Technology. All computations presented in this article were obtained using one CPU.

CI0504871