Automated Drawing of Structural Molecular Formulas under Constraints

Patrick C. Fricker†

Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen (SCAI), Schloss Birlinghoven, 53757 Sankt Augustin, Germany

Marcus Gastreich

BioSolveIT GmbH, An der Ziegelei 75, 53757 Sankt Augustin, Germany

Matthias Rarey*

Center for Bioinformatics (ZBH), University of Hamburg, Bundesstrasse 43, 20146 Hamburg, Germany

Received January 23, 2004

In this paper, we present a new algorithm for automated drawing of 2D structural formulas of molecules. The algorithm is based on the classical scheme of a drawing queue placing the molecular fragments in a sequential way. We extend the concept of so-called prefabricated units developed for complex ring systems to automatically created drawing units for chains and rings which will then be assembled in a sequential fashion. The approach is fast and can be naturally extended to the problem of drawing molecules with common core structures. Further on, we present an algorithm that allows the drawing of 2D structural formulas under directional constraints assigned to a subset of bonds. Since no numerical optimization is necessary, the algorithm creates drawings of small organic molecules on the order of 500 structures per second. The new algorithm is relevant for all kinds of prediction and analysis software presenting a large number of probably similar molecular structures to the user of the software.

INTRODUCTION

Drawing two-dimensional structural formulas from molecular graphs or connection tables belongs to the earliest applications of computers in chemistry.[1] Chemical drawing programs are part of a chemist's standard software repertoire, most of which contain automated structure diagram generators. As the number of applications in molecular modeling dealing with large molecule sets grows, the need for automated structure diagram generators increases. All kinds of applications ranging from virtual screening—similarity- or structure-based—via de novo design and library design to high-throughput screening (HTS) data analysis and lead optimization approaches have to present molecular structures to the user in a convenient way to browse through at least tens to hundreds of structures in a short time.

The problem of automated structure diagram generation is a complex one with regard to both modeling the problem and the aspect of solving it. While covering all rules of drawing chemical structures is a manageable task, covering all rules describing the *esthetics* of molecular structures is already quite complex (see, for example, ref 2). Even if a complete model for drawing chemical structures is given, the drawing problem itself is a difficult combinatorial problem to solve. In fact, detecting whether there is a planar drawing of a graph with constant edge lengths is known to be an NP-hard problem.[3]

Several software tools for structure diagram generation exist, most of which are semiautomatic in the sense that they clean up a hand-drawn structure. Basically every chemical drawing program has some kind of cleanup functionality; examples are ISIS/Draw[4] or ChemDraw.[5] Well-known software tools for fully automatic structure diagram generation are Daylight's depict,[6] the CACTVS system,[7] or JChemPaint,[8] an open source initiative for structure diagram generation. A comprehensive review on models and algorithms for structure diagram generation can be found in ref 9.

Under the viewpoint of the above-mentioned software applications in the area of molecular modeling and chem-informatics, the problem of automated structure diagram generation presents an additional aspect for consideration. Usually, the molecules of interest show a relationship to each other: In combinatorial libraries, there are common core structures, in docking or de novo design, the compounds are assumed to all fit into a protein active site of interest, in virtual screening, the molecules searched should be similar to a given query molecule, and, in HTS analysis, their mutual similarity leads to the formation of clusters.

For an efficient analysis of such compound sets, it is not sufficient to create a structure diagram for each individual molecule. Rather it is necessary to draw the molecules in such a way that the mutual relationships become visible to the modeler. In the case of combinatorial libraries, common substructures should be drawn and oriented in the same way. In fact, this should be a straightforward extension to most algorithms for structure diagram generation. The situation becomes more complicated if molecular similarities beyond

* Corresponding author phone: +49 40 42838 7351; fax: +49 40 42838 7352; e-mail: rarey@zbh.uni-hamburg.de.

† Current address: Center for Bioinformatics (ZBH), University of Hamburg, Bundesstrasse 43, 20146 Hamburg, Germany.

common substructures are to be represented in the drawing. To our knowledge, the only published work dealing with this problem can be found in ref 10. The authors describe an algorithm that identifies a common superstructure during the drawing of individual molecular structures.

In this paper, we present a novel algorithm that is able to deal with directional constraints, that is, constraints requiring compounds to be drawn in certain orientations, fragments in specified directions, etc. These constraints may be generated either manually or automatically by a software tool. We first present the overall drawing algorithm that is the basis of the constraint drawing procedure. This algorithm is closely related to existing methods published earlier.[9] We then explain extensions for drawing combinatorial libraries and for drawing under directional constraints. Finally, we show some examples of structure diagrams for related molecules created by the software.

METHODS

Automated Structure Diagram Generation (SDG). A well-known approach for automated structure diagram generation is the successive construction of a layout under utilization of a so-called drawing queue.[9] The molecule is divided into small pieces termed drawing units. An initial piece, the seed, is selected and drawn. All pieces which are adjacent to the seed are added to the drawing queue. In an iterative process, pieces are taken from the drawing queue, the layout is extended by the extracted piece, and neighbor pieces are added to the drawing queue. Within this overall concept for SDG, several degrees of freedom remain in the algorithm design, most importantly what exactly the drawing units are, how the seed is selected, how local design decisions are made, how to avoid or resolve overlapping moieties, and how to draw complex ring systems. For most of these issues, good solutions have already been developed and are summarized in ref 9. Since our approach for unconstrained drawing basically follows the lines of existing approaches, we will give only a rough outline here. We will then describe our extensions for drawing under constraints in more detail afterward.

Basically, the molecule is divided into drawing units, more precisely, the so-called chain- and ring-drawing units. In contrast to drawing acyclic parts atom by atom, the novel algorithm searches for longest chains, which are then drawn within a single step. This strategy favors elongated layouts since bends are only introduced if necessary for resolving overlap. The subdivision into drawing units is done with standard graph algorithm techniques.[11] The subdivision algorithm considers *E*/*Z* configurations at double and amide bonds such that a chain-drawing unit can be drawn in a zigzag layout without violating stereochemical features. Figure 1 gives an example of such a subdivision.

The drawing algorithm starts with the longest chain-drawing unit found in the molecule. To draw a chain, a zigzag layout starting at the connection to a previously drawn unit with alternating ±30° around the main-chain direction is created. To draw a ring-drawing unit, the ring system is further subdivided into the smallest set of shortest cycles with standard algorithms.[12] Starting from the connecting bond, rings are successively attached until the whole ring system is drawn. Rings are drawn in the order of increasing

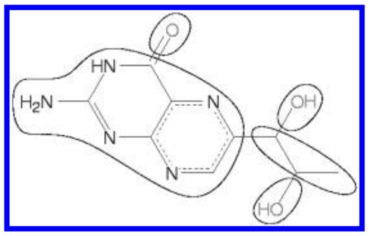

Figure 1. A molecule (biopterin) divided into drawing units. The drawing units for this molecule are encircled. Ring-drawing units always contain a "starting bond", which is the bond to the amino group in this example.

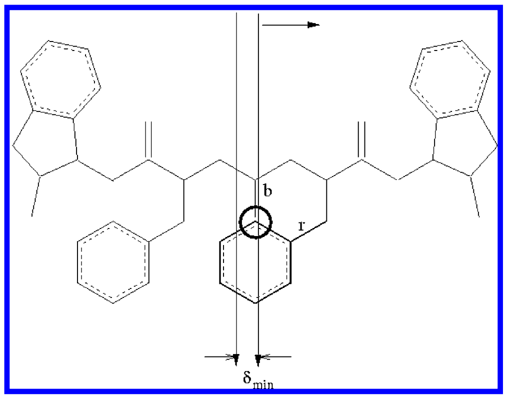

Figure 2. Collision detection with plain sweep. The sweep line runs from left to right and tests for the minimal atom distance. The collision of ring r and bond b is highlighted by a circle. Atom type labels are omitted for clarity.

number of not yet drawn bonds. Most ring systems can be drawn in reasonable quality with this strategy. For macrocycles, a special algorithm that allows nonconvex cycle drawing depending on adjacent rings is needed. Such an algorithm as well as the handling of prefabricated rings for highly bridged structures (e.g., cubanes, morphines, etc.) has not yet been implemented.

Whenever another drawing unit extends the layout, the added unit is checked for overlap with the previously drawn ones. Collisions are detected with a plane-sweep algorithm illustrated in Figure 2.

In case a collision occurs, the following strategy for modifying the layout is applied (see also Figure 3): First, all drawing units between the two colliding ones are identified; a modification has to be done in at least one of these units to resolve the collision. In contrast to previously published approaches, the algorithm first tries to resolve the collision by rotating complete drawing units with preservation of *E*/*Z* isomerism (a). In case atoms with multiple outgoing bonds occur, the order of outgoing bonds (b) and the angle pattern (c) at the atom can be modified. If this fails, selective bonds to terminal atoms are shortened (d) and rotations are performed within chain-drawing units (e). The final option tried is selective bond stretching in the middle of the path between the colliding atoms (f). To adhere to current standards in molecule diagrams, variations of bond lengths are kept to a minimum. For esthetical as well as technical reasons related to constrained drawing described later, our algorithm does *not* consider the distortion of angular patterns at atoms. In our experience, these rules represent the best compromise between collision removal and readability of the resulting diagrams.

Once the structure has been completely drawn, the overall structure is oriented such that the largest drawing unit lies

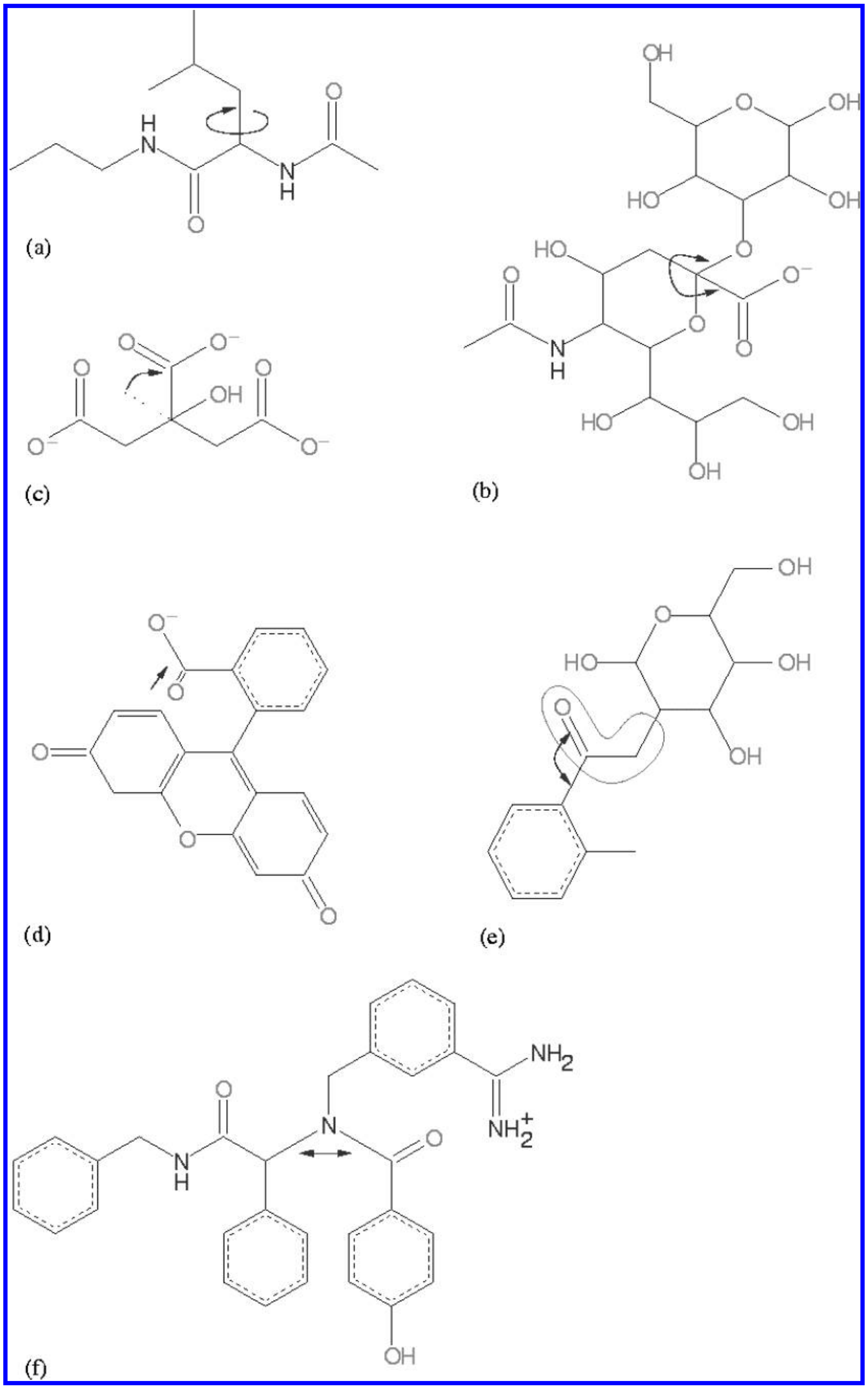

Figure 3. Different options for collision handling. Arrows indicate the operations that were performed for resolving conflicts.

horizontally and most substituents are pointing up or—in cases such as steroids—the standard layout for the ring system is used. Finally, the layout is decorated with atom names and stereochemical features.

Structure Diagram Generation for Combinatorial Libraries. If subsets of compounds of combinatorial libraries are to be drawn, the layouts are most useful if the common core structure is oriented in the same way for each drawn

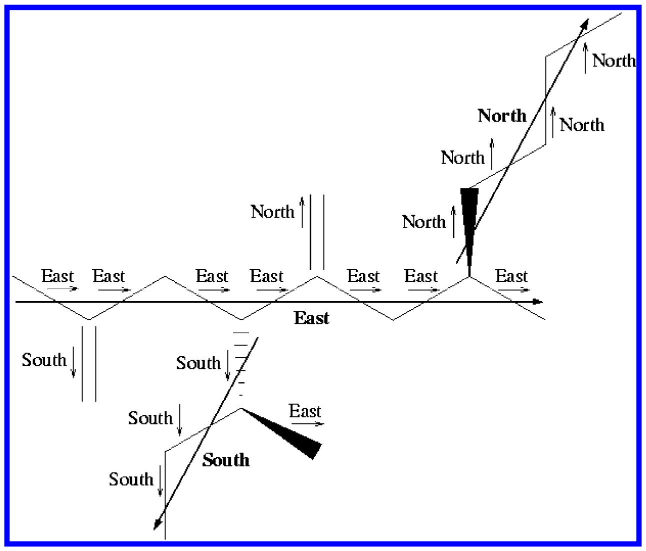

Figure 4. So-called logical directions are assigned to both chains (bold arrows) and individual bonds of the molecule skeleton. The direction of a bond is defined as the direction of the containing chain.

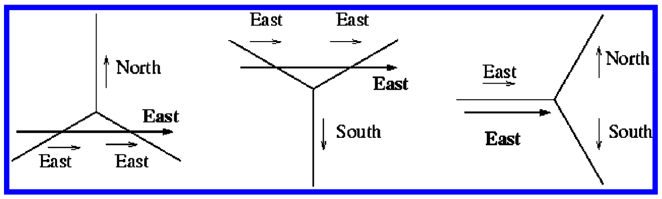

Figure 5. All drawing options for an atom with three drawn bonds, assuming that the main-chain direction is east.

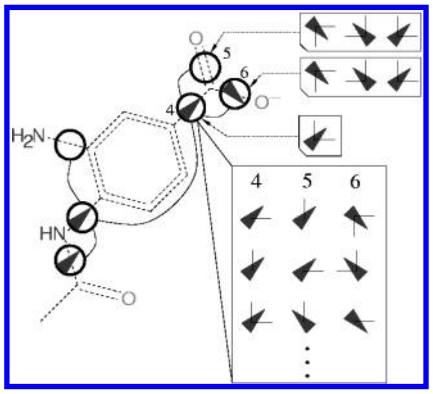

Figure 6. The direction tree represents possible drawings overlaid with its corresponding molecule. The nodes represent single bonds or ring systems. Arrows indicate the predefined direction of the constrained bond (the directionality of the bond is not shown). In the box below, exemplary rules stored at node 4 are shown. Every rule contains a direction (arrow) and possible zigzag states (lines) for node 4 itself and subsequent nodes 5 and 6. The orientations left for nodes 4−6 after the incorporation of the user constraint for node 4 are shown in the upper right corner.

molecule. This task can be solved with the above-described SDG algorithm with minor modifications.

In the first phase, a layout is created for the core structure. This is done by selecting the library molecule with the largest substituents and drawing it as described above. From this drawing, all layout information belonging to the common core structure is extracted. The rationale for this heuristic is that, with high probability, the core layout for the largest compound can be transferred to all other compounds.

In the second phase, each molecule of the library is drawn separately in the following way: First the layout of the common core is created using the information from the first phase. For every substituent leaving the core structure, the substituent itself as well as the direction it leaves the core structure is added to the drawing queue. The SDG algorithm can then be applied as before with some exceptions for collision handling: Since all core instances should be kept in a common orientation, only bond stretching and shrinking are allowed for collision handling within the core structures.

Dealing with Directional Constraints. A more challenging task is the creation of consistent layouts for similar molecules without a significant common substructure. The problem occurs in all kinds of data resulting from computer-aided molecular design applications. To separate the layout problem from the application or from a specific similarity measure, we developed a system for describing directional constraints in layouts. To avoid being overspecific, we use eight so-called "logical" directions (north, northeast, east, southeast, etc.) describing the orientation of a drawing unit. Note that, due to the zigzag drawing of chains, bonds of a chain show an alternate twist to the right and left with respect to the chain direction (see Figure 4). To describe the orientation of a bond, we extend the concept of logical directions by a so-called zigzag state, which is either *z*-right (right turn with respect to the chain-main direction) or *z*-left (left turn). In the case of a ring-drawing unit, we assign logical directions to the outgoing chains closest to the direction of corresponding outgoing bonds. The advantage of this system is that one can specify the rough direction of a functional group without hindering the layout algorithm too much as would be the case if exact angles were specified.

While the drawing algorithm proceeds, an orientation can be assigned to each acyclic bond. Drawing rules can be described by the notion of orientations. As an example, Figure 5 shows all possible ways of drawing an atom with three bonds for the direction east within the system of logical directions.

We now assume that a user or a program has specified logical directions for a subset of bonds. These bonds will be called user-constrained. Our algorithm for creating a layout fulfilling these user-defined directions proceeds as follows.

During the first phase, the algorithm creates a so-called direction tree. The tree contains a node for each acyclic bond lying between user-constrained bonds. The root of the tree is an arbitrarily selected node representing a layout-constrained bond; the directed edges of the tree follow the molecular graph pointing away from the root. Nodes are called ring nodes if they represent a bond pointing to a ring system and chain nodes otherwise. For each node, we will store a set of local drawing rules; each drawing rule is represented by a vector of orientations, one orientation for the node itself and one or more orientations for each child node with respect to the tree. These local drawing rules can now be created as follows: For a chain node, the set of possible orientations are enumerated on the basis of the chosen layout scheme (see Figure 5). For a ring node, the ring system is drawn and the relative angles of all outgoing bonds are used to create rule sets by mapping directions to their closest logical representative and combining them combinatorially with possible zigzag states. Figure 6 gives an example for a direction tree of a molecule.

In the second phase, the algorithm traverses the direction tree to find all orientations (direction + zigzag state) for each bond as well as all drawing rules that are consistent with the set of user-defined directions. These orientations and drawing rules are called valid; all others being inconsistent with the user-defined directions are called invalid. The traversal starts at the root and proceeds in a breadth-first manner, keeping all nodes in one of the sets P (processed), Q (queued for processing), and U (untouched). For each node, a list of possible orientations as well as a list of valid local drawing rules is maintained. During the traversal, the algorithm performs the following steps:

```
direction_tree_traversal (direction tree nodes N)
  • Select starting node s
  • P ← ∅, Q ← {s}, U ← N \ {s}
  • While Q is not empty do:
      • remove the first node v within Q and put it into P
      • calculate the local drawing rules for node v consistent with all valid orientations stored at v
      • for each child s from v do:
          o remove s from set U and put it into Q
          o calculate the valid orientations based on the rules of the node v
      • for each child s from v do:
          o remove all orientations at s and all drawing rules at v inconsistent with user-constraints
          o if no orientation at s is left, stop (drawing within given user constraints is impossible)
          o in case that orientations were removed due to user-constraints, call the procedure tightening_up (v) (see below)
```

When *direction_tree_traversal* finishes, each node in the direction tree contains a set of valid orientations that conform to all user constraints.

The task of the tightening procedure is to remove all local drawing rules and orientations that became invalid due to removed orientations at neighboring nodes. The tightening procedure traverses recursively all nodes of the direction tree stored in P and Q. It is based on three functions called *tightening*, *tightening_down*, and *tightening_up*. *tightening* itself performs the removal of invalid orientations and drawing rules, and *tightening_down* performs tightening in a parent-to-child manner and *tightening_up* correspondingly in a child-to-parent manner. The functions perform the following steps:

```
tightening (v)
  • remove all invalid drawing rules at v: drawing rules became invalid because orientations were previously removed for v or its children
  • remove all orientations in v and its children which are not used by the remaining drawing rules at v
tightening_down (v)
  • if v ∈ P (i.e. v is already processed) then call tightening (v)
  • for each child c in which tightening has removed orientations call tightening_down(c)
tightening_up (v)
  • call tightening(v)
  • if tightening has removed orientations in v then call tightening_up (parent(v))
  • for each child c in which tightening has removed orientations call tightening_down(c)
```

When tightening is finished, all invalid rules and orientations are removed. To show this, two cases have to be considered. Let us first assume we are processing in parent-to-child manner using the *tightening_down* function. *tightening_down*(v) was called because orientations were removed from v while parent(v) was processed. Let R(v) be the removed orientations. Since v had a valid drawing rule set and orientation set beforehand, *tightening_down*(v) will at most remove drawing rules using orientations from R(v) for v. This implies that no additional orientations for v will be removed by *tightening_down*(v) and all drawing rules at parent(v) stay valid. Let us now assume we are processing in a child-to-parent manner using the *tightening_up* function. The same argument as in the previous case shows that all rules and orientations at child(v) stay valid during the processing of v by *tightening_up*. The above argument also makes clear that, during the whole tightening procedure for a user constraint, each node is visited only once.

In the third phase, the calculated orientations are converted into a structure diagram using a drawing routine very similar to the one explained above for unconstrained structure diagram generation. The diagram is created iteratively; in each iteration, a drawing unit is added to the existing diagram and the overlap is removed. The drawing unit selected is the longest chain of bonds which can be drawn into the same direction. To determine this longest chain, the direction tree can be searched for the longest chain of bonds allowed to be drawn into the same direction with alternate zigzag states. During this search, the tightening function must be applied again to verify that whenever an orientation is selected, all remaining bonds can still be drawn.

Two problems remain during structure diagram generation under constraints; both of them have been solved heuristically so far: When bonds with *E*/*Z* isomerism occur, certain combinations of directions at neighboring bonds are forbidden. Our drawing algorithm considers *E*/*Z* isomerism; however, this is done in a greedy way by just selecting an arbitrary valid combination of directions. It should be noted that *E*/*Z* isomerism could in principle be modeled similar to drawing constraints for ring systems and therefore be considered during the creation of the direction tree. The second problem is to deal with possible collisions during drawing. As drawing with constraints contains the SDG problem as a special case, it cannot be expected to solve the overall problem efficiently. We therefore use a simple backtracking procedure that redraws larger and larger parts with alternate decisions on drawing directions.

Apart from collision handling, the presented algorithm always finds a valid drawing if it exists. However, if the user constraints cannot be obeyed, it might be desirable to find a drawing with the minimal number of violated user constraints. Even further, a weighting scheme to differentiate between more and less important constraints would be useful.

So far, the optimization variant of the SDG problem described above has only been solved heuristically by modifying or even ignoring incompatible user constraints on-the-fly. With this extension, the order of the direction-tree traversal affects the calculated drawings. Thus, a priority queue is used as the underlying data structure for Q, favoring nodes pointing to important constraints: The priority of a node in the direction tree is defined as the sum of the weights of all constraints in the corresponding subtree. In addition,

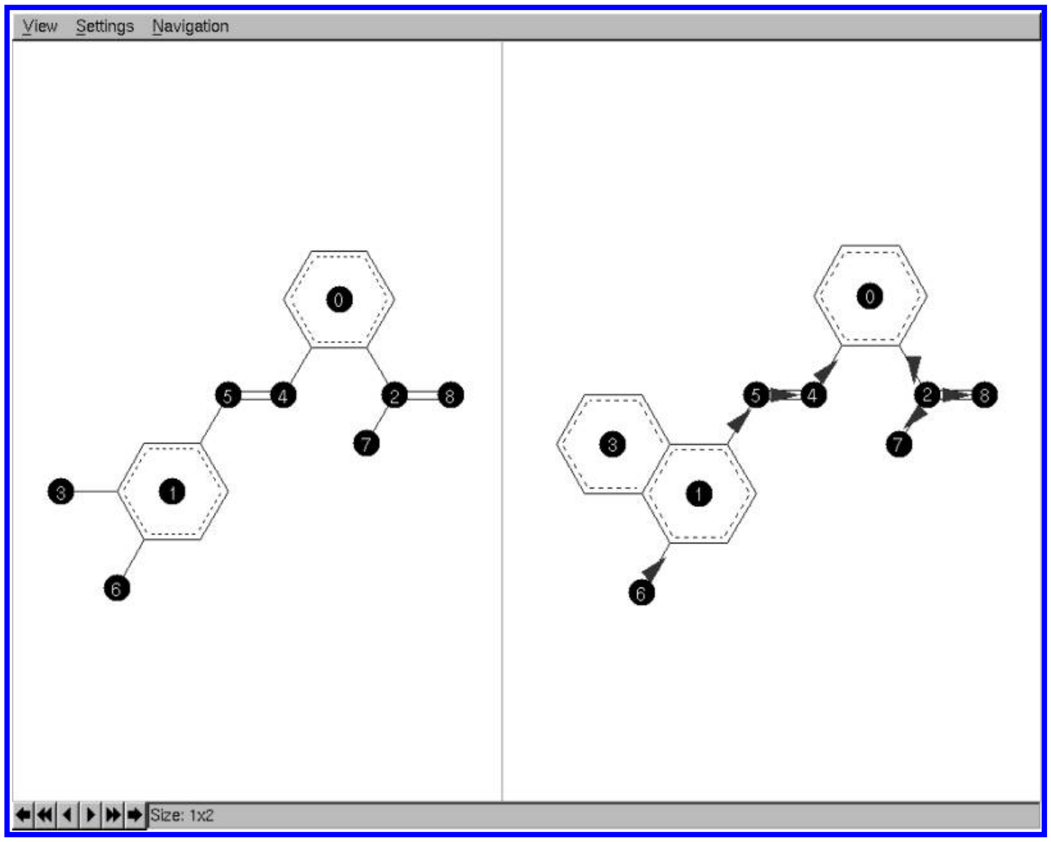

Figure 7. A Feature Tree matching for two similar molecules and the corresponding drawing constraints. The Feature Tree nodes are represented by circles. The numbers represent matches created during the Feature Tree comparison. From neighboring matched nodes (for example, match 0 and 4), directional constraints for connecting bonds are derived for the right molecule by copying the bond directions from the template molecule (left). For clarity, the atom labels (for example, OH at node 7) were removed.

all end-standing nodes with constraints are in turn considered as roots.

Feature Tree-Based Creation of Directional Constraints. Directional constraints are supposed to be applicable to different types of user constraints. One such user constraint may be to draw compounds in such a way that their mutual similarity is reflected in the diagram. In the following, we will discuss the application of directional constraints based on the Feature Tree similarity measure.[13]

In short, a Feature Tree is a molecular, topology-preserving descriptor; molecules are represented as trees, the nodes of which are assigned with various physicochemical properties. The result of a comparison of two molecules represented by Feature Trees is a list of matches relating subtrees of roughly the same size and chemical properties. In other words, matches assign similar fragments under consideration of their relative arrangement within the molecules to each other. (Note that, due to the merging of rings into single Feature Tree nodes, the relative arrangement of functional groups within a ring is not considered in the similarity calculation and therefore not considered in the drawing constraints.) The approach allows the comparison of molecules with low computer resources; computing times lie on the order of milliseconds. An example of a Feature Tree matching is shown in Figure 7.

The following procedure is applied to create layouts reflecting the Feature Tree similarity for a given set of molecules: First, an arbitrary molecule is selected as a template and drawn without constraints. Depending on the application, molecules with a certain role (e.g., the query compound in virtual screening or a cluster center in data analysis) are chosen. Then, for each molecule of the set, the matches of a Feature Tree comparison to the template molecule are converted into directional constraints.

For this translation step, bond paths between matches, bond paths within Feature Tree matches, and directions implied by the relative arrangement of rings in ring systems are considered. For the bond paths and ring systems, directions from the template are taken and used as the user constraints for the corresponding bond paths in the molecule to be drawn. An example of a translation of a Feature Tree matching into directional constraints is shown in Figure 7.

RESULTS AND DISCUSSION

In this section, we will first cover the basic application of converting topological molecular data to structure diagrams, discussing speed and limitations of the software. In subsequent paragraphs, the application of constraints to drawings as outlined in the previous section shall be assessed.

Automated Structure Diagram Generation. To estimate the performance of the drawing routines, we have applied the algorithm to the NCI cancer library.[14] This database contains 122473 compounds in the 2003 version.

In Figure 8 we have drawn 15 randomly selected compounds of differing complexity as examples. Avoiding hard disk write access leads to a computing time of 4.5 min and a memory usage of less than 1 MB on a standard PC (Intel P4, 2.6 GHz, 1 GB main memory running under Linux). During this time, the program was successful in converting

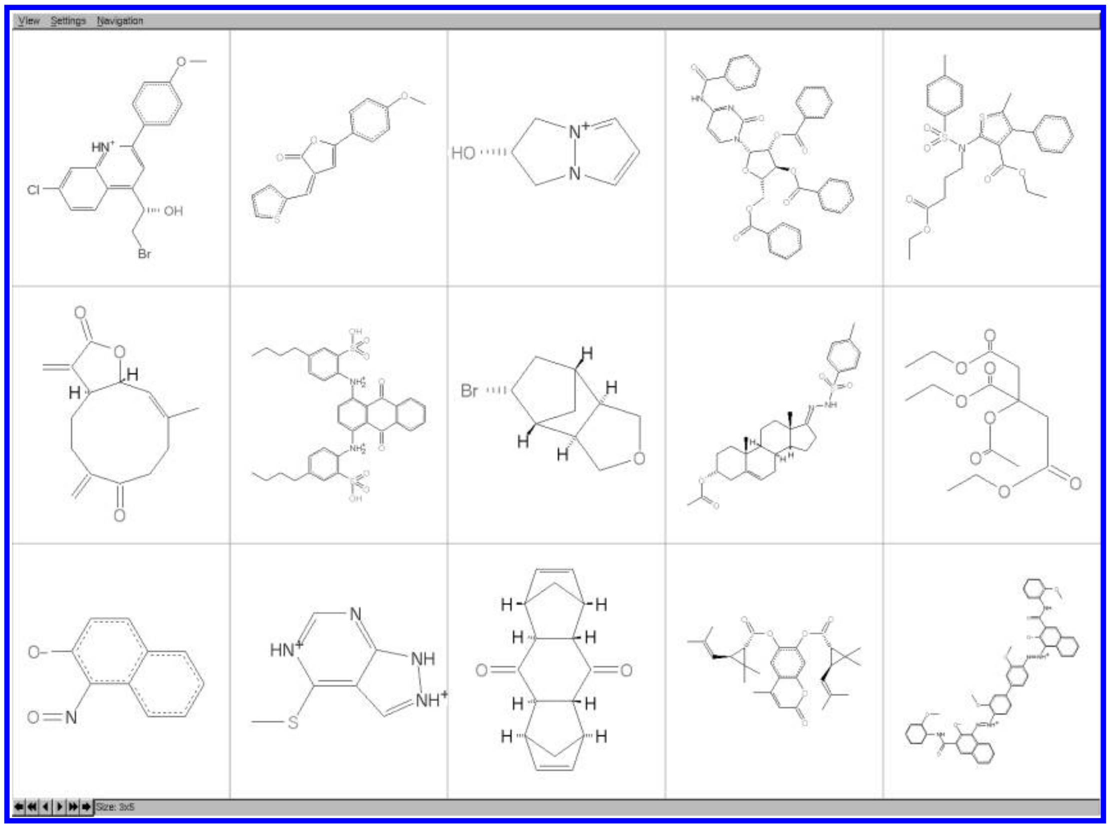

Figure 8. Fifteen molecules from the NCI Cancer Library (random selection). The structure diagrams are of differing complexity; compound number 10 (row 2, column 5) is an example for collision treatment by bond stretching. Note that, throughout this paper, all stereo annotations are drawn in a perspective manner from the stereo center, as recommended in ref 2.

Table 1. Statistics of Failures for Drawing the NCI Cancer Database[a]

	count	percentage
total in database	122473	100.00
valid compounds (no salts, no single atoms)	121972	99.59
failures		
ring topology too complex	2748	2.24
collisions	3814	3.11
coordination number larger than 5	243	0.20
other	45	0.04
failures total	6850	5.59

[a] Generating a compound requires 2 ms on average.

roughly 95% of the molecules into structure diagrams. This corresponds to an average drawing speed of 2 ms per compound. The user may have several hundred compounds on the screen per second and thus obtain a fast visual impression of structural properties of compounds when scrolling through relatively large libraries.

For a total of 6850 molecules, no drawings could be generated—corresponding to 5.6% of all compounds. The main reasons for failure are distributed as shown in Table 1.

It becomes obvious that collisions and rings are the problems that are responsible for most of the failures. This is common to most of the SDG programs and basically due to the constraint of all-equal bond lengths and the complexity of nontrivial bridged cyclic compounds as they occur, for example, in the morphines. It will be left for future work to overcome the aforementioned difficulties. Here, though, we only want to highlight two obvious paths to solutions: For ring systems which are too complex, one would define a library of drawn ring systems to be accessed as it is done in other SDG algorithms. For collisions, the described heuristic for collision handling should be replaced by a more sophisticated optimization procedure. Carrying more than five substituents on one atom usually occurs in organometallic compounds, which rarely occur among small drug molecules as they are considered here.

Combinatorial Libraries. Often, combinatorial libraries are made up of a common central fragment, the core, to which R groups, i.e., lists of alternate substituents, are attached. As mentioned above, when these libraries are dealt with, it is desirable to draw a list of compounds generated from the library description in such a way that only the changes in R-groups are reflected in the drawing, i.e., that the core is preserved in the layout.

A frequently used chemical reaction in combinatorial chemistry is the Ugi reaction.[15] We applied the new SDG algorithm to compounds generated from the combinatorial docking module FlexX-C[16] using an Ugi reaction-based library.[16,17]

In Figure 9, a selection of 28 compounds meant to represent certain diversity within the library are displayed. For better visibility, we have highlighted the core fragment of the library. All core fragments could be drawn meeting the constraint derived from the layout of the first drawn molecule (see Figure 9).

The Ugi library also shows the advantage of bond stretching compared to angle distortion. Bond stretching is used to remove collisions in all compounds of the second and fourth rows without destroying the overall similar drawing of all compounds too much.

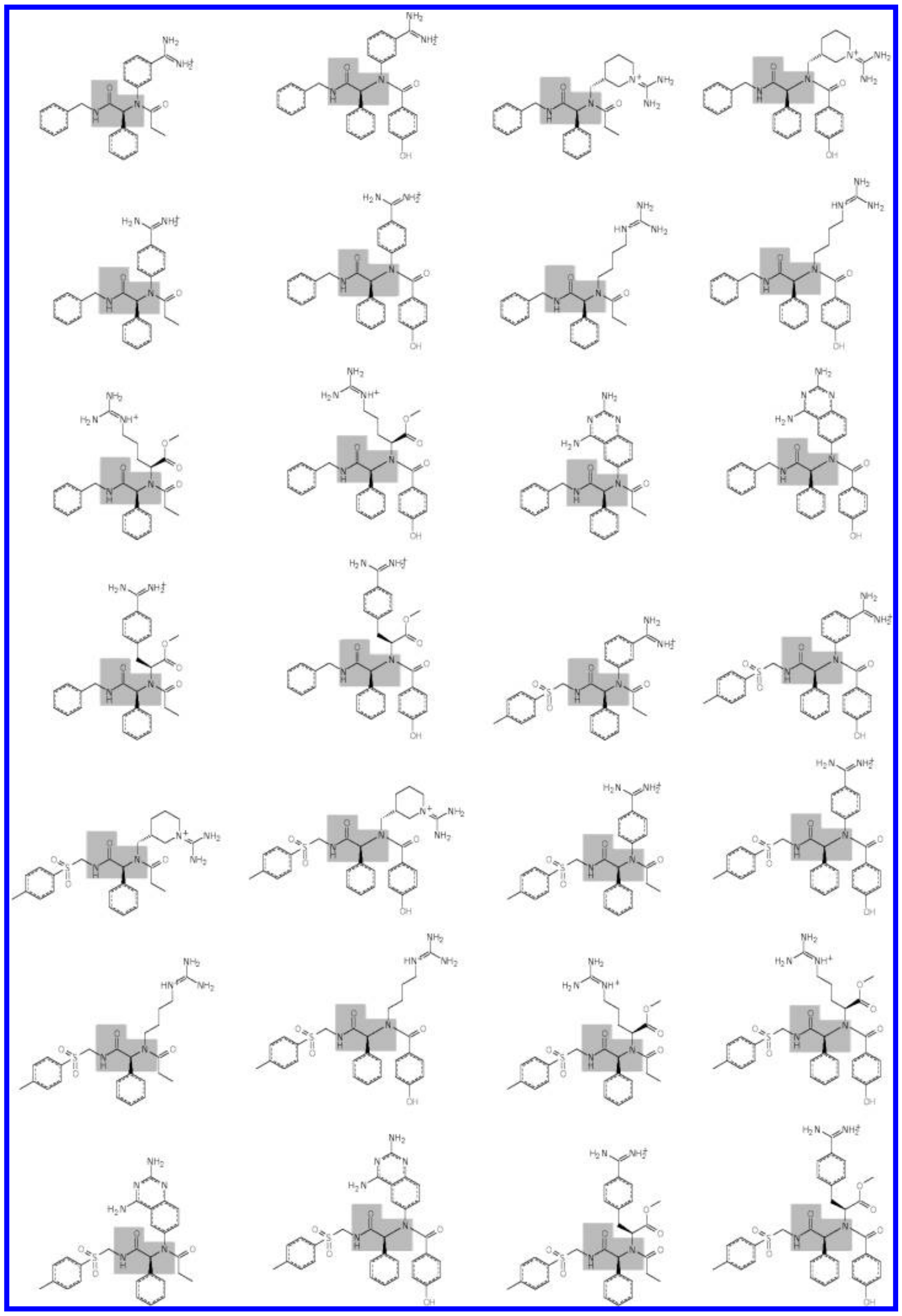

Figure 9. Twenty-eight selected molecules from a combinatorial library based on the Ugi reaction. The core fragment of the library has been highlighted to emphasize the all-equal orientation resulting from the constraint that cores should be drawn in a similar way.

Constrained Automated Drawing I: Manually Imposing Constraints. The pharmacophore centers of angiotensin-converting enzyme (ACE) inhibitors are well-known—despite the fact that the target 3D structure has only recently been published.[18] The pharmacophore consists of three functions: a carboxylic acid group, a metal-complexing group, and a carbonyl group. Mostly, the metal-chelating groups are sulfur-containing groups such as a mercapto group or simply carboxylate groups. In an example scenario, we utilized 28 ACE inhibitors from ref 19. Figures 10 and 11 show both the unconstrained output of our program and the results obtained under manually constraining the bonds. The latter have been chosen such that the outcomes demonstrate the similarity of the compounds with respect to their pharmacophore centers. The molecule in the upper left has been considered as the reference compound. The task is now asfollows: Given constraints reflecting the similarity of the compounds with respect to the pharmacophore centers, the program shall generate appropriate structure diagrams. In fact, the ACE dataset is already a quite challenging example for creating a consistent set of structure diagrams. The pharmacophore groups are connected by different ring systems such that they cannot be drawn in exactly the same orientation. Another problem with this example is the underspecification: In the molecules in row 3, column 1, and row 4, column 4, the leftmost "east" constraints are not enough; to improve the result, a fourth constraint for the phenyl moiety is needed. For illustration, this was only done for the second molecule. Two other examples for which there is too much freedom for drawing the molecule are shown in

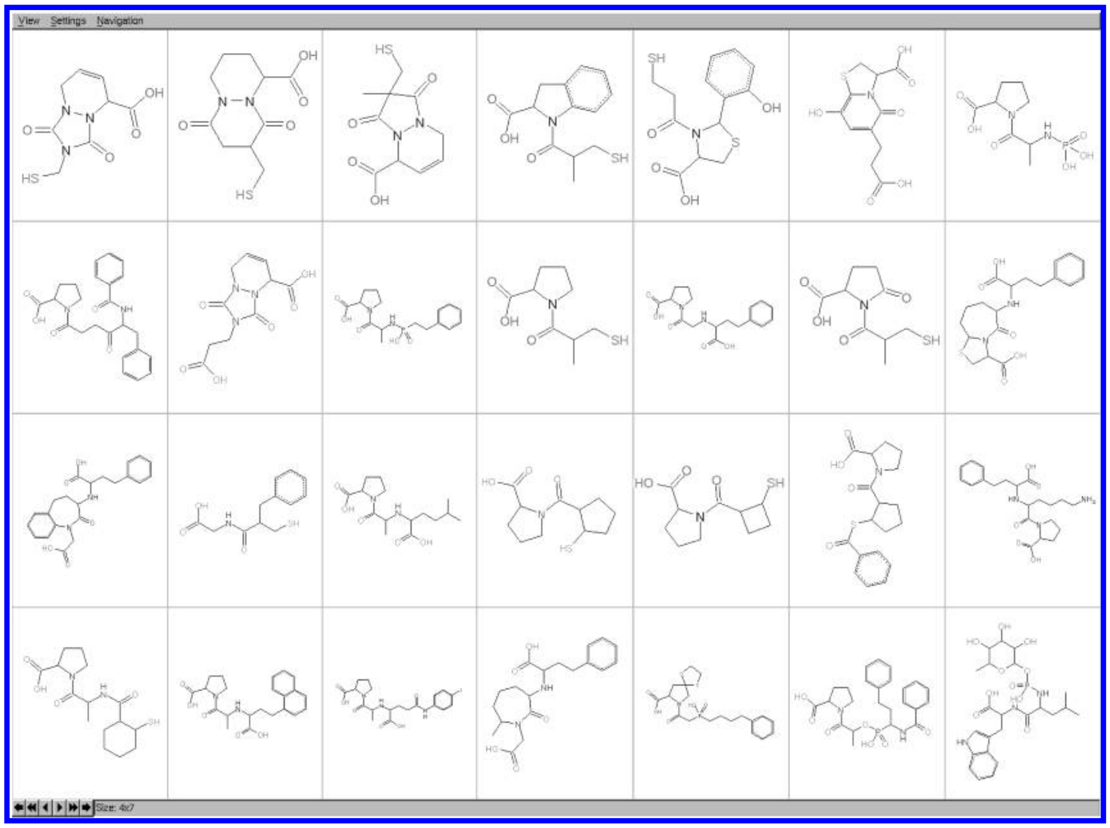

Figure 10. Automatically generated structure diagrams for 28 known ACE inhibitors.

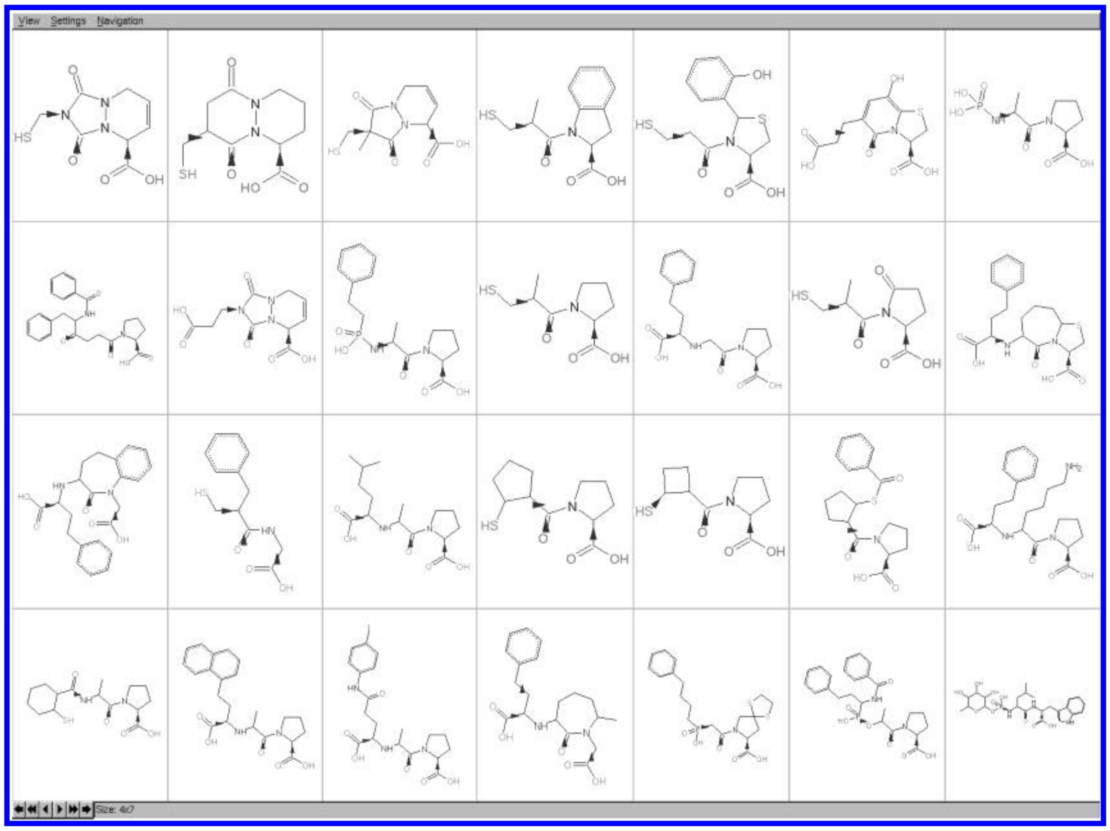

Figure 11. Output after *manual* adjustment of the constraints to reflect the preservation of the pharmacophore points. The pharmacophore pattern is preserved throughout the structure diagrams, and nontrivial functional groups for the pharmacophore become much more obvious. The arrows indicate the imposed constraints to be met.

row 3, columns 4 and 5. If the constraint is set directly at the mercapto group, the more specific "northeast" is necessary to get the desired result. This was done for the second molecule. With east for this bond, an alternative layout can be found, as shown in Figure 12. Likewise, to find the "expected" layout for the molecule in row 3, column 6, an

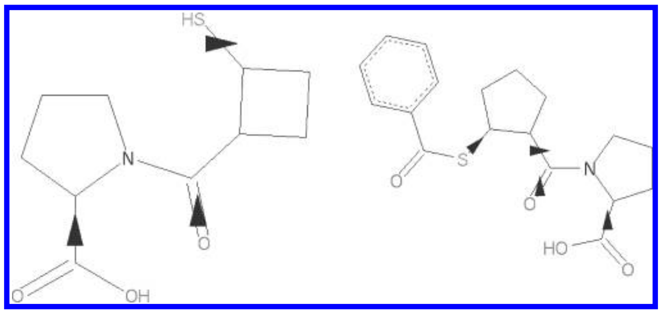

Figure 12. For the first molecule, there is more than one way to fulfill the constraints. The molecule is drawn in an alternative way also fulfilling all constraints (compare Figure 11, row 3, column 5). On the right, an alternative drawing after addition of an additional constraint is shown (compare Figure 11, row 3, column 6).

additional constraint is needed (also shown in Figure 12). In most cases, however, the SDG algorithm creates layouts such that the pharmacophore pattern becomes visible.

Constrained Automated Drawing II: Imposing Constraints from Feature Tree Similarity Comparisons. In this section, we take one step further to the *automatic* generation of constraints: As outlined above, a possible way to automatically derive constraints is to make use of Feature Tree matches. Such matches represent *pairwise* respective pharmacophore groups; for example, the OH group of one compound may match the mercapto functionality of another. Utilizing this information as a constraint for drawing leads to layouts from which the user can more easily reestablish, e.g., the similarity in the arrangement of pharmacophore groups.

As an application scenario for this rationale, we have chosen two compound classes, followed by a “query versus hit list” scenario. The first of the two compound classes is made of benzodiazepines, whereas the second group represents tricyclic neuroleptics, both taken from ref 20. All respective structures were available in SD file format. When drawing is done freely, that is, compoundwise one by one without any mutual relation of the compounds being considered, the common and different features are difficult to capture visually (see Figure 13).

On the other hand, if Feature Tree similarity is used to constrain the drawings, we can align the molecules; the resulting output is captured from the screen in Figure 14. We have arbitrarily chosen the compound exhibiting the highest number of rotatable bonds as the reference.

Similarly, the results for the tricyclic neuroleptics are displayed in Figures 15 and 16. For both sets, the runtime is lower than 10 ms per compound for both the Feature Tree comparison and the drawing with constraints combined. The drawing with constraints alone needs one-third of the time. As the molecules are relatively simple, the structure diagram generation without constraints needs less than 1 ms per compound in this case.

One has to be aware of the fact that there is a certain freedom in choosing the template or reference compound, and the outcomes may differ because of the pairwise character of the Feature Tree comparisons. In principle, it can happen that two compounds similar to each other but dissimilar to the reference compound are drawn differently because the pairwise similarity was not detected via the two comparisons to the reference. Drawing with respect to a selected compound can be desirable, for example, in virtual screening. In principle, also a *set* of compounds could be used to extract directional constraints by “overlaying” a *manifold* of Feature Trees, thus forming a so-called model.[21] Then, this model could be used as a less biased guiding reference for subsequent constrained drawing.

An application scenario closely connected to the aforementioned one results from de novo molecular design: Starting with a query molecule, it is desirable to have a 2D display of the top ranks of a hit list reflecting their similarity to the query. To this end, we refer to a recent paper.[22] In this paper, the authors have exemplarily taken a virtual screening result for dopamine D4 antagonists at varying Feature Tree similarity levels. It has been deducted that 0.9 reflects a value which still is sensibly understandable by visual inspection by a human. The six compounds selected at this similarity level have been drawn in both an unconstrained (Figure 17) and a constrained (Figure 18) way; the query itself served as template.

Certainly, all recent scenarios may be altered by employing another “constraint generator”, since the implementation of the SDG algorithm is generic. The two examples are meant both to give an idea of the usefulness of the drawing engine and to document a proof of concept.

CONCLUSIONS

In this work we presented a novel algorithm for automated structure diagram generation. The algorithm is fast and deals with most chemical structures relevant in drug design. Application to the NCI cancer database led to a success rate of 94.4%—resulting in an average speed of 2 ms per structure diagram generation. The main focus of the new method is on how to generate sets of structure diagrams for *related* compounds, a problem which frequently arises in computer-aided molecular design. In one variant, we considered combinatorial libraries with a common core structure. The algorithm creates structure diagrams with a common orientation and layout of the core heuristically optimized by taking several molecules of the library into account. The second variant considered a more generic problem of drawing under user-given directional constraints. In the first example reported, a combinatorial library with a common core structure was drawn. The common core structure and the related R groups of the molecules of the library could be easily recognized in the drawn structures. The benefit of the novel approach becomes even more obvious when applied to sets of molecules showing similarities on a level of pharmacophore features rather than on the substructure level. We therefore applied the structure generator to the output of similarity calculations performed with Feature Trees. On the basis of the Feature Tree comparison algorithm, a consistent set of directional constraints was created which could then be used for structure diagram generation. In a second example, several compounds similar to a given query were drawn in a way that parts of molecules that are considered similar by Feature Trees are visualized through the layout of the structures. In a third example, a chemistry space search was performed. The newly created molecular structures were, as above, drawn in a way that the mutual similarities were clearly visible.

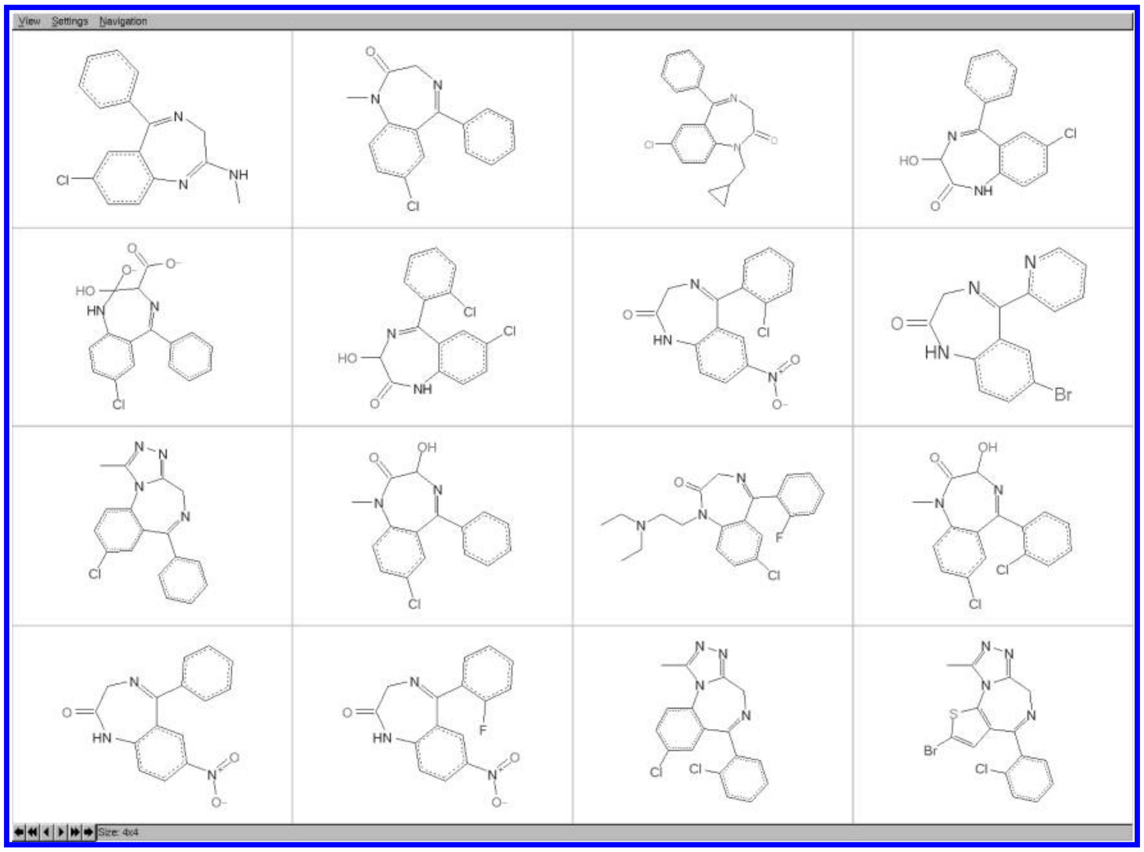

Figure 13. Benzodiazepines drawn without constraints. The orientations of the compounds do not correlate; for compound 5 (Tranxilium, counted from left to right and top to bottom), only the anion of the potassium salt has been drawn.

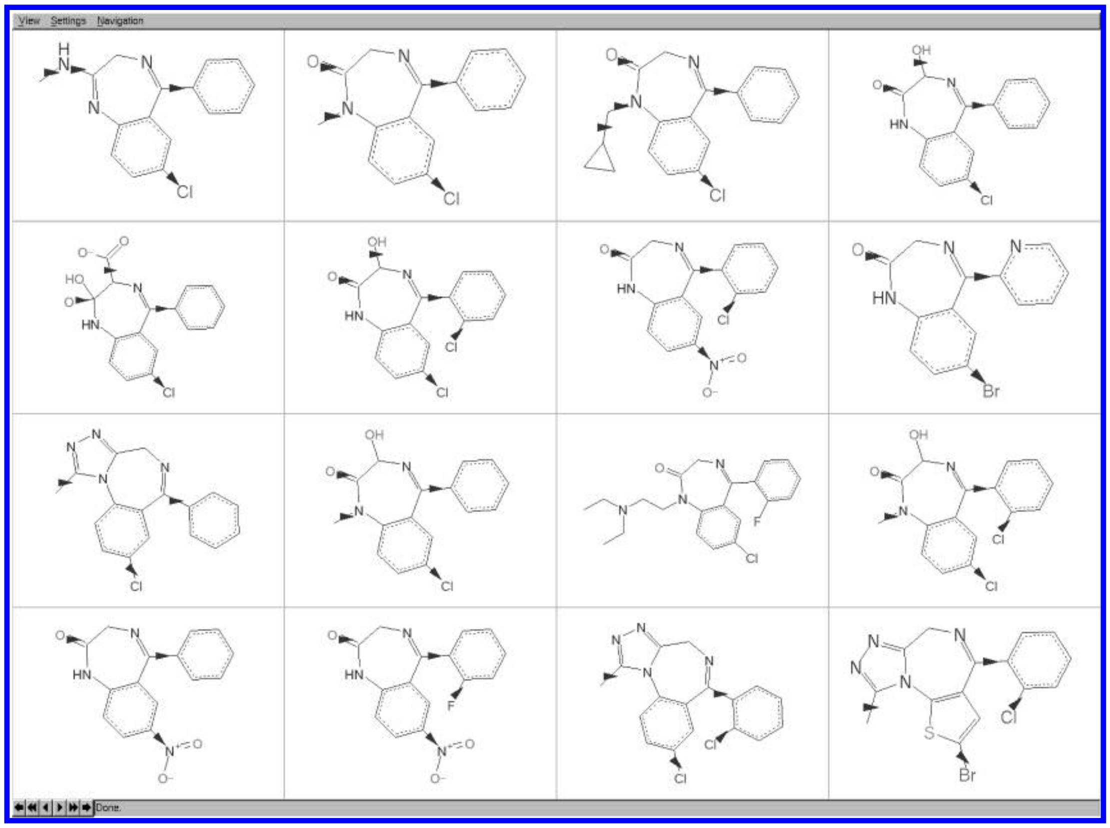

Figure 14. Benzodiazepines drawn as above with compound 11 (row 3, column 3) as the reference. All others have been drawn with the constraint of being aligned according to Feature Tree matches with respect to compound 11.

During the process of solving this problem, molecular similarities have to be converted into a set of directional constraints. While this renders the software quite flexible in dealing with all kinds of similarity measures, it is also clear that the conversion from similarity to directional constraints might be difficult to achieve. The situation becomes espe-

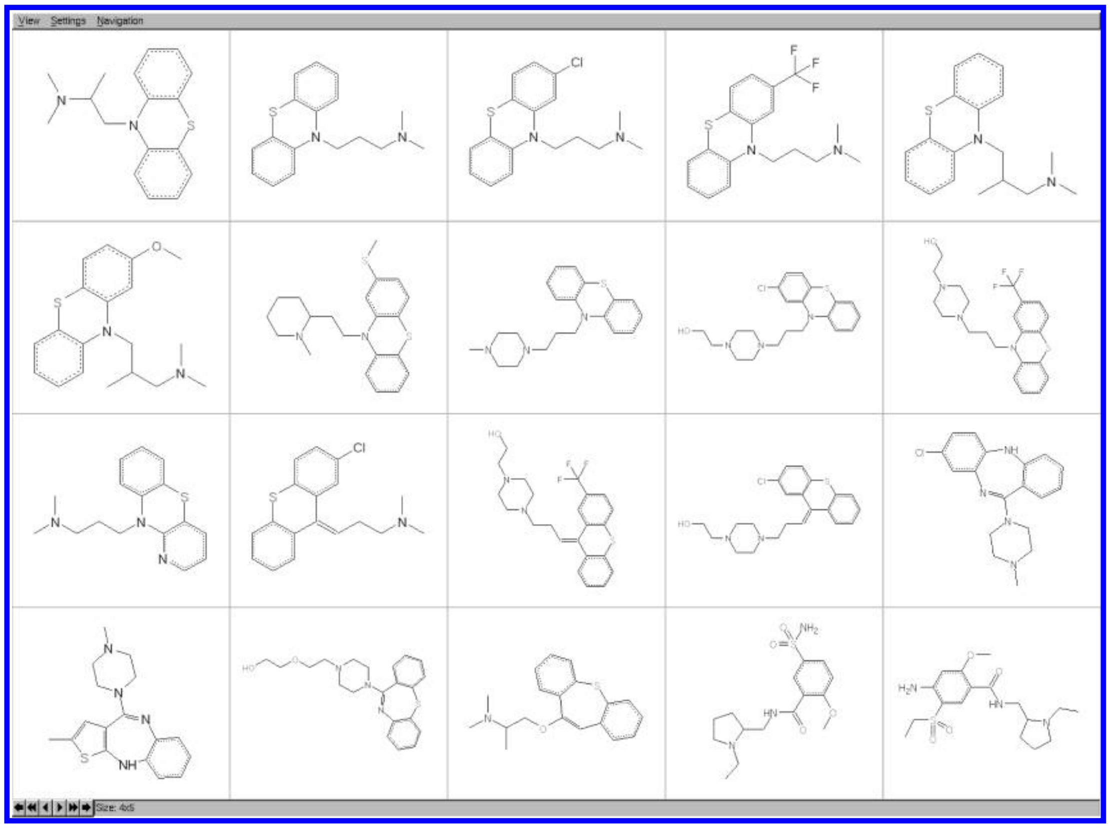

Figure 15. Tricyclic neuroleptics, unconstrained drawing. There is no mutual correlation with respect to orientation of the structure diagrams.

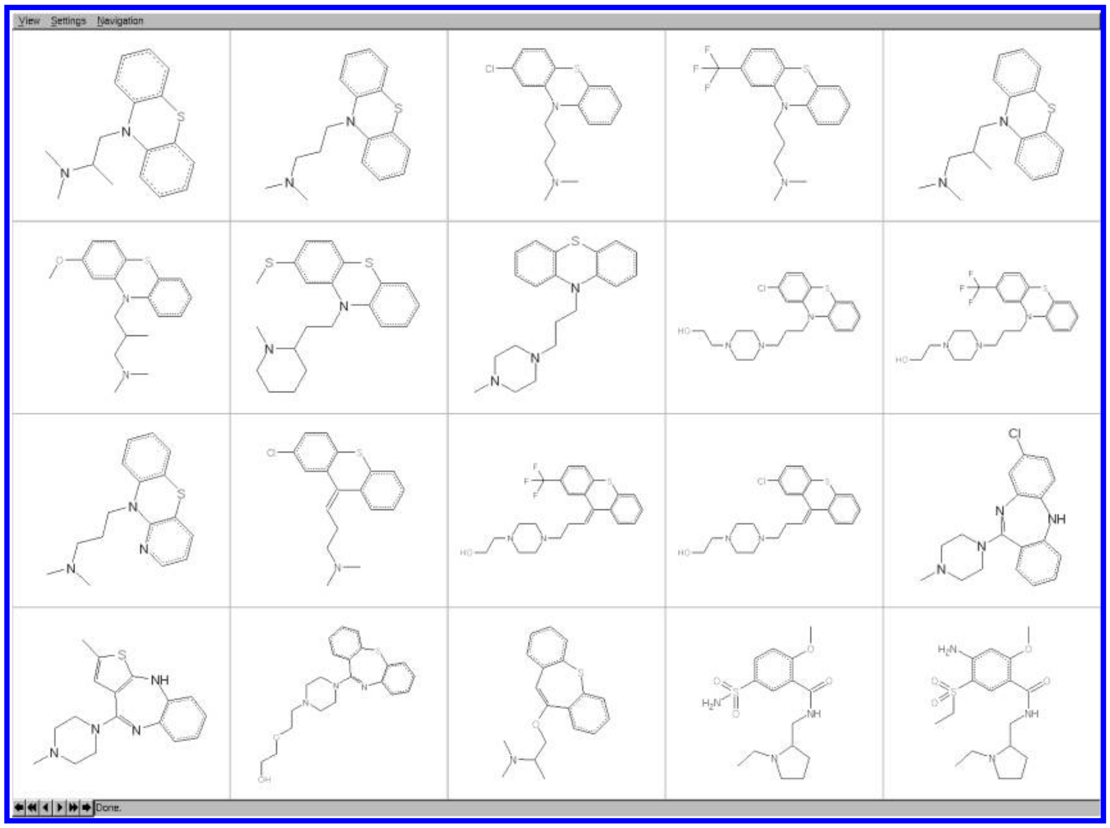

Figure 16. Tricyclic neuroleptics drawn under Feature Tree constraints as above using the compound in row 2, column 4, as reference. If it exists, the tricyclic moiety is preserved to be oriented in the upper right part of the respective square. If it does not, the user obtains fragments that—on the Feature Tree level—align to the tricyclic fragment, for example, the compound in row 4, column 4, at the respective site. For reasons of better visibility, no arrows indicating the constraints have been drawn here.

cially complicated whenever linear descriptors such as structural keys or hashed fingerprints are used. In these cases, it remains to be answered how to break down an overall similarity value to a mapping of the molecules such that directional constraints can be created. Here, we have demonstrated how an alternative approach (Feature Trees)

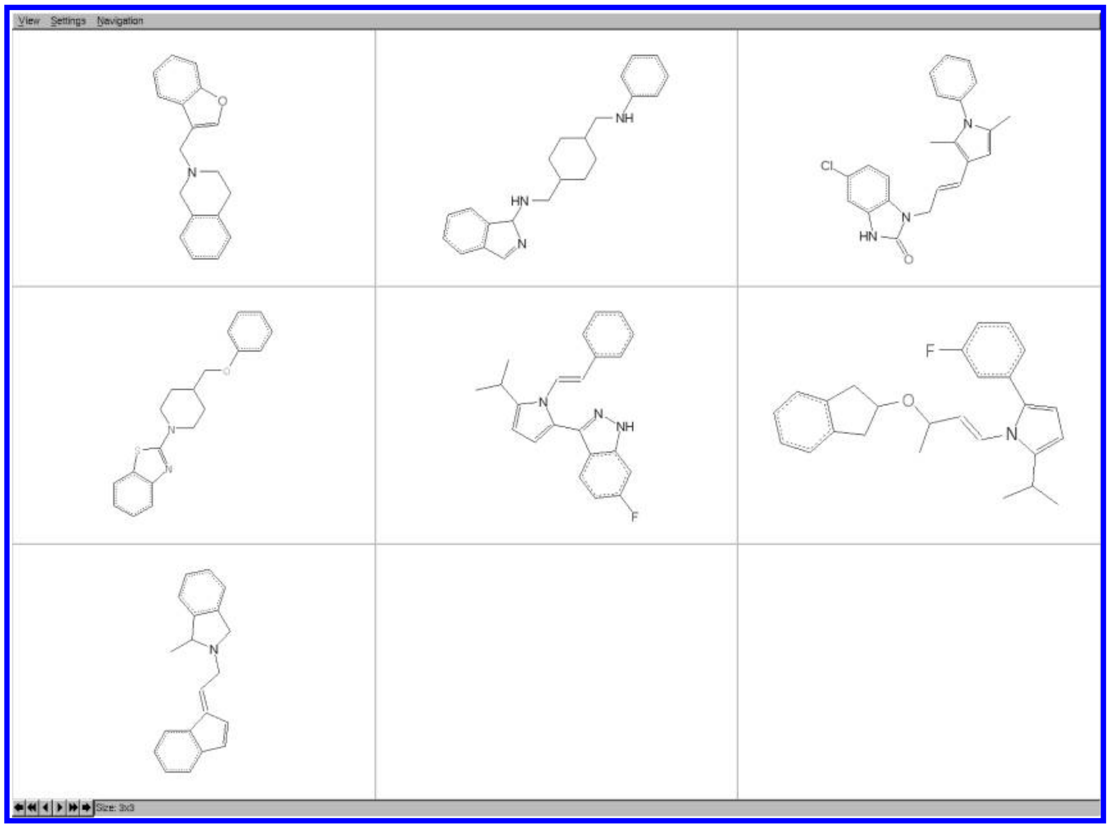

Figure 17. Dopamine D4 antagonist predictions drawn without constraints.

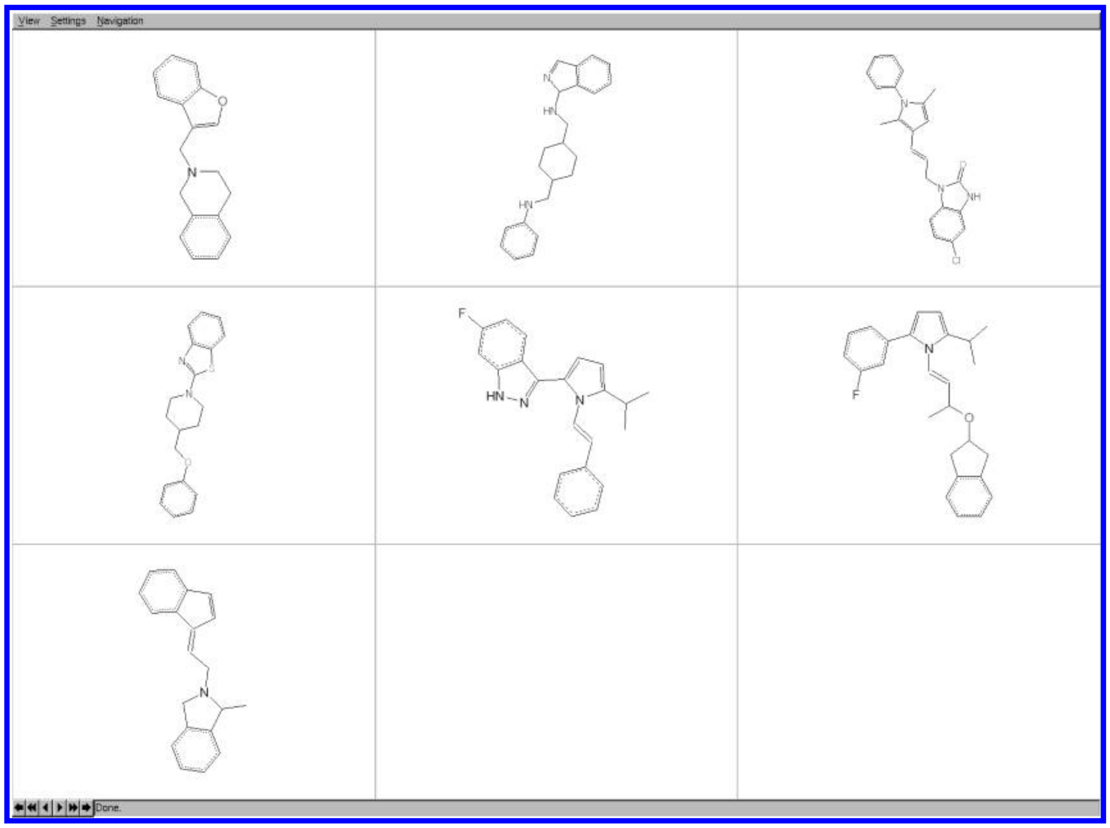

Figure 18. Dopamine D4 antagonist predictions drawn under the constraint to align to the query compound (row 1, column 1). The alignment has been automatically carried out in Feature Tree similarity space. The query compound serves as the template and is drawn in the upper left corner.

can create mappings between molecules from which directional constraints could be deduced. This rationale should be directly transferable to other structural-alignment-based approaches.

SOFTWARE AVAILABILITY

Information on the ZBHs SDG activities is available at http://www.zbh.uni-hamburg.de/SDG. The software will be made available within HTSview by BioSolveIT GmbH, St. Augustin, Germany. Further information can be found via the Internet at http://www.biosolveit.de.

ACKNOWLEDGMENT

The authors express their gratitude to Hugo Kubinyi for valuable discussions and pointers to application scenarios.

REFERENCES AND NOTES

(1) Zimmerman, B. L. *Computerized-Generated Structural Formulas with Standard Ring Orientations*; University of Pennsylvania: Philadelphia, PA, 1971.

(2) Maehr, H. Graphic Representation of Configuration in Two-Dimensional Space. Current Conventions, Clarifications, and Proposed Extensions. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 984−902.

(3) Eades, P.; Wormald, N. Fixed edge length graph drawing is NP-hard. *Discrete Appl. Math.* **1990**, *28*, 111−134.

(4) ISIS/Draw: MDL Information Systems Inc., San Leandro, CA.

(5) ChemDraw: CambridgeSoft Corp., Cambridge, MA.

(6) depict: Daylight Chemical Information Systems Inc., Mission Viejo, CA.

(7) Ihlenfeld, W. D.; Takahashi, Y.; Abe, H.; Sasaki, S. Computation and Management of Chemical Properties in CACTVS: An extensible Networked Approach toward Modularity and Flexibility. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 109−116.

(8) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493−500.

(9) Helson, H. E. Structure Diagram Generation. In *Reviews in Computational Chemistry*; Lipkowitz, B., Boyd, D. B., Eds.; Wiley-VCH: New York, 1999; pp 313−398.

(10) Boissonnat, J. D.; Cazals, F.; Flötotto, J. 2D-Structure Drawings of Similar Molecules. *Graph Drawing* **2000**, 115−126.

(11) Corman, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. *Introduction to Algorithms*, 2nd ed.; MIT Press: Cambridge, MA, 2001.

(12) Balducci, P.; Perlman, R. S. Efficient exact solution of the ring perception problem. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 822−831.

(13) Rarey, M.; Dixon, J. S. Feature Trees: A new molecular similarity measure based on tree matching. *J. Comput.-Aided Mol. Des.* **1998**, *12*, 471−490.

(14) NCI Cancer Library. http://resresources.nci.nih.gov, National Cancer Institute, Bethesda, MD.

(15) Ugi, I.; Eberle, G.; Eckert, H.; Lagerlund, I.; Marquarding, D.; Skorna, G.; Urban, R.; Wackerle, L.; Zychlinski, H. v. The Present Status of Peptide Synthesis by Four-Component Condensation and Related Chemistry. In *Proceedings of the 5th American Peptide Symposium*; Goodman, M., Meienhofer, J., Eds.; Halsted Press, Wiley & Sons: New York, 1977; pp 484−487.

(16) Rarey, M.; Lengauer, T. A Recursive Algorithm for Efficient Combinatorial Library Docking. *Perspect. Drug Discovery Des.* **2000**, *20*, 63−81.

(17) Weber, L.; Wallbaum, S.; Broger, C.; Gubernator, K. Optimization of the Biological Activity of Combinatorial Compound Libraries by a Genetic Algorithm. *Angew. Chem., Int. Ed.* **1995**, *34*, 2280−2282.

(18) Natesh, R.; Schwager, S. L. U.; Sturrock, E. D.; Acharya, K. R. Crystal structure of the human angiotensin-converting enzyme-lisinopril complex. *Nature* **2003**, *421*, 551−554.

(19) Mayer, D.; Naylor, C. B.; Motoc, I.; Marshall, G. R. A unique geometry of the active site of angiotensin-converting enzyme consistent with structure−activity studies. *J. Comput.-Aided Mol. Des.* **1987**, *1*, 3−16.

(20) Mutschler, E.; Geisslinger, G.; Kroemer, H. K.; Schäfer-Korting, M. *Mutschler Arzneimittelwirkungen*, 8th ed.; Wissenschaftliche Verlagsgesellschaft mbH: Stuttgart, Germany, 2001.

(21) Zimmermann, M.; Rarey, M.; Naumann, T.; Matter, H.; Hessler, G.; Lengauer, T. Extracting knowledge from high-throughput screening data: towards the generation of biophore models. In *14th EuroQSAR 2002 Symposium*; Ford, M., et al., Eds.; Blackwell Publishing: Bournemouth, U.K., 2002; pp 63−67.

(22) Rarey, M.; Stahl, M. Similarity Searching in Large Combinatorial Chemistry Spaces. *J. Comput.-Aided Mol. Des.* **2001**, *15*, 497−520.

CI049958U