Computer-Assisted Mechanistic Evaluation of Organic Reactions. 25.[1] Structure Diagram Positioning

Harold E. Helson* and William L. Jorgensen*

Department of Chemistry, Yale University, New Haven, Connecticut 06511

Received February 15, 1994*

As part of a procedure to improve the appearance of structure diagrams produced by the CAMEO program, a technique has been developed for distributing molecules evenly about the computer's display. There are two stages: first, the molecules are located strategically in complementary, noninterfering positions; second, they are dynamically scattered using a force field-like approach. A "jumping heuristic" solves some difficult fits.

I. INTRODUCTION

One of the problems encountered by any chemical synthesis program, in which the output molecules arise by the creation and deletion of bonds, is that of redrawing the new molecules so they are neat and attractive. A second, distinct problem is how to position the molecules so that they do not overlap and are evenly distributed about the display. This subject does not appear to have been broached before in the chemical literature.[2] The "repositioning" problem has proved deceptively difficult for us to solve.[3] However, a swift and effective procedure has been developed, as described here.

In our first implementation, there was simply a list of *ca.* nine positions on the screen to which molecules were dispatched in an arbitrary order. There was no provision for molecules beyond this count. All molecules were scaled down until any overlap between adjacent molecules disappeared. It was common, particularly when a large molecule was situated at the screen's edge or next to another large molecule, for the structure to be scaled down to ludicrous proportions. This and other defects led us to seek a new approach. A good repositioning algorithm should do the following:

(1) be fast
(2) keep molecules as large as possible
(3) maintain a minimum distance between all objects, including molecules and screen edges
(4) distribute empty space equally, so that the void separating adjacent molecules is consistent

The new treatment consists of two independent, sequentially executed lines of attack, which each have distinct advantages and weaknesses. The first approach is analytical in that a deterministic, noniterative procedure leads to a disposition of molecules that is largely unrelated to their starting positions. The second approach is an iterative procedure whereby the final arrangement is visibly related to the original. It is referred to as "dynamic" repositioning because the molecules are treated like charged particles that repel one another, giving rise to a dynamic migration of positions toward an arrangement of lower "potential". The analytic method is fast and is effective in setting molecules in good relative positions, *i.e.* making molecules fit together so that the least scaling down, or the most scaling up, is possible. It may, however, produce large, wasteful spaces. By contrast, the dynamic method is relatively slow and may not improve a bad starting arrangement but is effective at nudging molecules toward empty spaces. We used the analytic procedure alone for several years with satisfactory results. With the dynamic procedure as a tail-end supplement, the results are markedly better.

While the two methods are normally used in tandem, a useful application of dynamic repositioning alone is to position a new molecule in a field of other molecules without disturbing the latter. For example, a molecule created by CAMEO's name translation facility can be added directly to the screen, provided there is room for it, thus avoiding a distracting screen refresh. The new molecule is roughly situated in the largest hole and allowed to drift away from the closest points of contact. To achieve this effect, the preexisting molecules are assigned the special "static" attribute, making them immovable but nonetheless repelling other objects.

A common feature of both algorithms is that every molecule is represented as the smallest rectangle that will enclose it. As such the molecules are referred to as "boxes". The dimensions and area of any box are therefore dependent on the underlying molecule's orientation. A margin is added around each box, effectively increasing its molecular size. No matter how difficult the system of boxes, this padding ensures that a certain margin of free space will be left around every molecule. A little extra space is allocated around isolated (nonbonded) atoms in order that they may be distinguished more easily. Margins complicate the analytic algorithm substantially because they destory proportionality; *e.g.*, they do not contribute to the boxes' expansion during scaling operations.

II. ANALYTIC REPOSITIONING ALGORITHM

The analytic algorithm consists of four stages (Algorithm I). In the first, the boxes are assigned relative positions such that the least scaling down is likely to be required in order to avert overlap. (Note that the molecules may be too big initially not to encroach on one another. Redrawing, in particular, confers a standard bond length, so that if many or large molecules are present, some will necessarily overlap, as dramatized in Figure 1). Vertical and horizontal scaling components arises as a byproduct of box positioning.

In the second stage, a lateral scaling factor is calculated. In the third stage, the actual factor is determined that would bring the molecules just short of coming into contact. Scaling is executed if scaling down is necessary or if scaling up is significant. In the final stage the wall-hugging boxes are moved toward the center of the screen. It is this step that actually puts space between molecules; hitherto they have been packed together. Following is a detailed discussion of the algorithm.

* Abstract published in *Advance ACS Abstracts*, June 15, 1994.

0095-2338/94/1634-0962$04.50/0

Algorithm I. Analytic Reposition Algorithm

(I) Create V and H groups and calculate V and H scaling factors (Figure 3)
- (1) Create boxes to describe molecules.
- (2) If only one molecule is present, process it separately: scale, translate to center, and exit.
- (3) Rank boxes by aspect.
- (4) Assign each box to V or H. Transfer middle box until the V and H scaling factors are as similar as possible.
- (5) Place the V-boxes to the right, flush bottom; place the H-boxes to the top, flush left.

(II) Calculate the lateral scaling factor and Unified Tangent
- (1) If only nonbonded atoms are present, for the Unified Tangent use the diagonal (45°) line that runs through the screen center; skip to part IV.
- (2) Calculate the Unified Tangent and lateral scaling factor (Figure 4).
  - (a) Calculate the V tangent lines.
  - (b) Calculate the H tangent lines.
  - (c) For each of V and H, construct an average tangent line.
  - (d) Calculate the Unified Tangent direction.
  - (e) Determine the pair of V- and H-boxes that protrude the most with respect to the Unified Tangent direction.
  - (f) Determine the separation between the V and H tangents.
  - (g) Calculate the V and H scaling susceptibilities.
    - (i) Find the pairs of V and H "reduced" protrusion points: the most protruding boxes when box margins are ignored.
    - (ii) The V (or H) scaling susceptibility is the component of the V (or H) reduced protrusion point vector in the direction normal to the unified tangent (eq 1).
    - (iii) Compensate each susceptibility for an imbalance in the number of V- and H-boxes, and provide a minimal value when only nonbonded atoms are involved.
  - (h) Calculate the lateral scaling factor (eq 2).
  - (i) Calculate the Unified Tangent (eq 3).

(III) Scale diagram if necessary
- (1) The scaling factor is the smallest of the H, V, and lateral scaling factors. Skip to part IV if the scaling factor is greater than, but close to, unity.
- (2) Scale original structure.
- (3) Remake box representations.
- (4) Redeploy boxes to their V and H locations.

(IV) Relax boxes toward the Unified Tangent

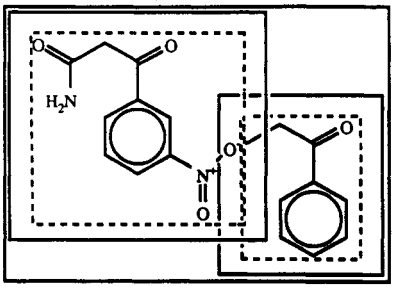

Figure 1. Lateral screen overlap.

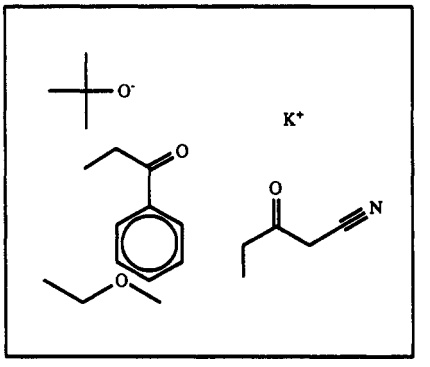

Figure 2. Example structure before repositioning.

A. V and H Assignments and Scaling Factors. The goal of the first stage is to label all boxes as either **Vertical** (tall) or **Horizontal** (**wide**) and to arrange them in a characteistic pattern that tends to minimize overlap. The N boxes are ranked according to their aspect, *i.e.* height to width ratio. Nonbonded atoms are given an aspect of one. Initially, the $N/2$ boxes with the largest aspects (tallest) are assigned to the Vertical group and the rest to the Horizontal group.

Conceptually, the **V-boxes** are placed side by side along the lower screen border, in order of decreasing aspect from right to left. The **H-boxes** are thought of as residing along the left screen border, in order of increasing aspect from top to bottom. Shortly, the boxes actually will be arranged by this theme, but it is easiest to imagine them so now. Thus, for example, the structures in Figure 2 are boxed and reassembled as in Figure 3.

The division into V and H groups, and the arrangement of those groups, is the key strategic element in the analytic repositioning algorithm. The molecules' chances for overlap are minimized because large molecules tend to be concentrated at opposite corners of the screen; as the V and H groups converge toward the bottom left, they get smaller, with the tendency that overlap is less likely.

Next, the effect of shifting the middle box to the opposite side (V or H) is assessed as follows. Let VSum be the sum of the widths of the V-boxes and HSum be the sum of the heights of the H-boxes (Figure 3).[4] Let **XScale** be the ratio of the full horizontal screen extent to VSum and **YScale** the ratio of the vertical screen extent to HSum. XScale and YScale are scaling factors that reveal how much room is free in either dimension. If the number of boxes is odd, the middle box is the $[(N + 1)/2]$th one (*e.g.*, the potassium of Figure 3) if, even, the innermost V-box (*e.g.* butoxide, if potassium were absent from Figure 3) is arbitrarily chosen. If transfer of the middle box to the opposite side would result in a decrease in $|XScale - YScale|$, the reassignment is accepted. Regardless, the test is repeated on whichever of the two neighboring boxes would bring (or approach) a reversal in sign of (XScale − YScale). The iterative procedure terminates when backtracking occurs; *i.e.* a previously tested box is reselected as the middle box. The result is to equalize the amounts that each side would have to be scaled in order just to fill the screen. Although the method is somewhat crude, because the other dimensions of the boxes are not considered, it simply and quickly optimizes the V–H partitioning.

Once the final V and H assignments have been made, the boxes are represented by extent rather than aspect: the V-boxes by height and the H-boxes by width. Often no change in order occurs, though in Figure 3 the ether molecule has moved. The new sequence is the more natural; ranking by aspect was

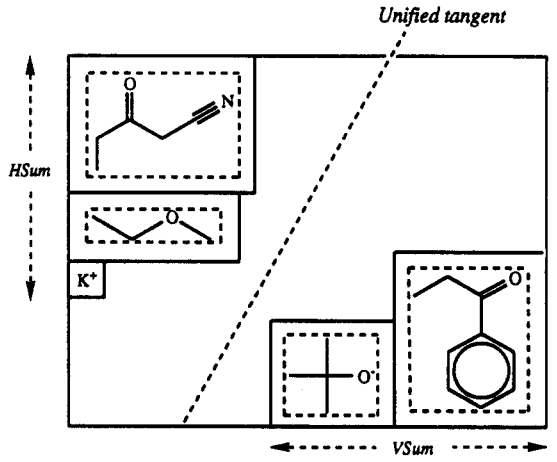

Figure 3. V and H assignments. V-boxes originate at the bottom right and H-boxes at the top left, and the two groups converge toward the bottom left. Solid lines indicate the positions of the added margins. The ether molecule was the top H-box after sorting by aspect, but came to the middle after sorting by width. The unified tangent will be calculated in later steps.

only necessary to ensure that the middle box transfer step involved the box whose dimensions were most suited to the opposite side. After sequencing, the boxes are at last placed in their described positions.

It may be that the V or H groups overflow the screen at this point, as indicated by a value of XScale or YScale less than unity. Most of the time, however, there is ample room in both dimensions, and the scaling factors are both greater than unity. In either case, we would scale the boxes (maintaining their relative positions and alignments) by an amount MIN-(XScale,YScale) in order to keep all boxes on screen but as large as possible.

It is also possible at this point for the boxes to overlap in the middle of the screen (Figure 1). The remainder of the algorithm relies on a roughly diagonal line called the **Unified Tangent** (Figure 3) that separates the V- from the H-boxes. The system will be scaled, if necessary, so that the V- and H-boxes are segregated on opposite sides of this line. This **lateral scaling factor** is the third and final value needed to determine the actual scaling factor, which will simply be the smallest of the three. Among this trio all modes of obstruction are prevented: XScale and YScale ensure that the boxes fit on screen, while the lateral factor ensures that boxes do not overlap. Considerable work is necessary, however, to obtain the Unified Tangent and the lateral scaling factor. This goal is pursued next.

B. Lateral Scaling Factor and Unified Tangent. The Unified Tangent is a roughly diagonal line midway between the V and H groups. It will be used both to ensure that boxes do not overlap and to provide a direction toward which boxes will migrate in the final stage. The object is to choose its angle and position so that the least scaling down is necessary (*i.e.* the lateral scaling factor is maximized). Its angle will be a weighted average of the **V- and H-tangents**, which are themselves the averages of the lines connecting the most protruding V- and H-boxes, respectively. This will become clearer by detailing and illustrating what is involved.

The V-tangents are those lines which intersect the (top left) corners of at least two V-boxes without passing through the bodies of any other V-boxes (Figure 4). Put metaphorically, if the boxes are viewed as steps in a staircase and a long wooden board is laid on them and seesawed, then a tangent occurs wherever the board is in contact with (at least) two steps.

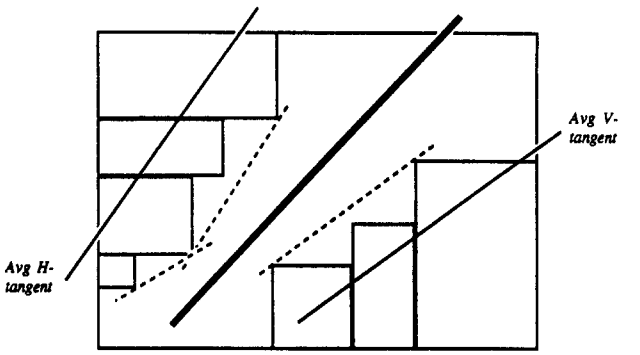

Figure 4. Calculation of the V- and H-tangents (dashed), V and H average tangents (solid), and unified tangent direction (heavy).

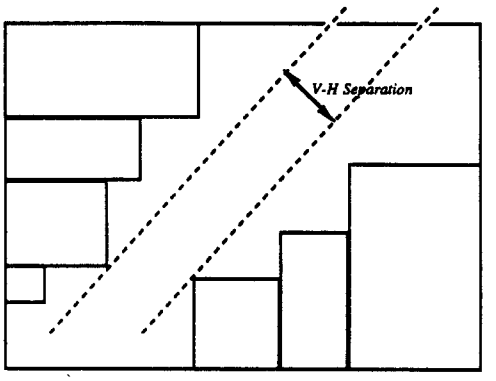

Figure 5. Unified tangent drawn through the V- and H-boxes that protrude most along the normal to this direction. The protruding boxes are not necessarily close to each other, as they are in this example.

Every V-tangent receives a weight equal to the distance between the (outermost) V-box corners that define the tangent.

There can be any number of tangents in principle, but most situations are trivial, having only zero or one. The H-tangents are found in an exactly analogous manner.

The **average V-tangent** is the weighted average of the individual V-tangent directions. It too receives a weight, for use in the next step, equal to the sum of the (margin-free) vertical extents of all the V-boxes. If there are no V-tangents (*i.e.* if there is at most one multiatomic V-box), an angle of 45° is used.

The average H-tangent is constructed similarly, and the **Unified Tangent direction** is obtained from the weighted average of the average V- and H-tangents (Figure 4).

Although elaborate, the above procedure gives a line that divides the two sides more or less evenly. A simple regression approach based on all the box corners might not work well because it would not properly emphasize interior corners. It is important to realize that although the Unified Tangent direction has been obtained, its position, for which a point through which the line passes is needed, has not yet been determined.

Next the distance between the V and H sides is calculated. The pair of most **protruding V- and H-boxes** is identified; these possess the innermost corners with respect to the normal to the Unified Tangent direction (Figure 5). The **V–H separation** is the gap between the two lines incident with the protrusion corners and represents the distance required to move one group before one of its boxes would collide with a member of the opposite group. (The actual free lateral distance between the V and H groups will be greater than this if the two protruding boxes do not happen to be adjacent. The value calculated, however, ensures that neither group crosses the Unified Tangent (calculated below). This is essential to the

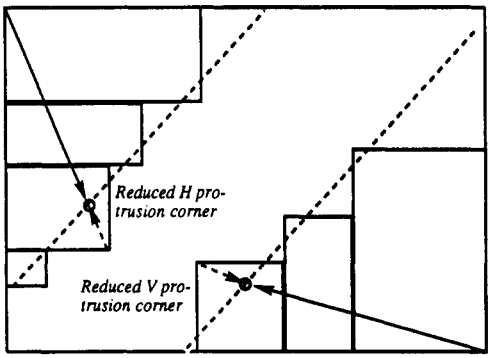

Figure 6. Reduced protrusion points, with lines parallel to the unified tangent passing through them. (Drawing not to scale.)

relaxation step, in which the two groups expand sheerwise along their own sides of this tangent.)

How the V–H separation is affected by scaling is complicated by the presence of the margin areas, which do not respond to scaling. To ascertain how each side will contribute, the pair of **reduced protrusion corners** is found. These are the most protruding corners when the boxes are all stripped of their empty margins (Figure 6). Which two boxes harbor the reduced protrusion corners are usually but not always the same as the ones found originally. The vector ($\bar{C}$) to a reduced protrusion point from the appropriate scren corner represents how the protruding box will move as a result of scaling the collection.

The susceptibility (of the V side, for example) to scaling is then

$$S_V = |0\ 0\ {}^{-}1| \times \bar{U}_N \cdot \bar{C}_V \qquad (1)$$

where $|0\ 0\ {}^{-}1| \times \bar{U}_N$ is the normal to the Unified Tangent (normalized) and $\bar{C}_V$ is the vector from the bottom right corner to the reduced V protrusion corner. S_V is called the **Vertical Scaling Susceptibility**. For example, if the system is scaled up, the amount by which the V–H separation is decreased is given by that component of $\bar{C}_V$ that lies in the direction of the V–H separator. An analogous procedure yields the Horizontal Scaling Susceptibility.

It frequently happens that one or the other susceptibility is quite small, because the boxes on that side represent nonbonded atoms or molecules small in size and number. This would cause the Unified Tangent to lie quite close to a corner, with the consequence that when the boxes are finally relaxed (*vide infra*), the ones on that side would scarcely move. Small susceptibilites are therefore augmented by a small amount porportional to the difference in the number of boxes on the two sides (1.2 character lengths per box) and are subject to a floor value (three character lengths). A side effect is to decrease the lateral scaling factor slightly from its proper value (see eq 2).

Lateral scaling should bring the V–H separation just to zero. It is found using the following relation:

$$\text{Lat\_ Scal\_ Fact} - 1.0 = (\text{V\_ H\_ Sepn})/(S_V + S_H) \qquad (2)$$

Both susceptibilities are positive, and if there is lateral overlap, then the V–H separation is negative and the scaling factor will be less than unity.

Finally, a point defining the Unified Tangent *position* is obtained. It is given by the unreduced protrusion point plus the susceptibility times the scaling increase:

$$\text{UT\_ Pnt} = P_{V,\text{red}} + (1.0 - \text{Lat\_ Scal\_ Fact}) \cdot \bar{S}_V \qquad (3)$$

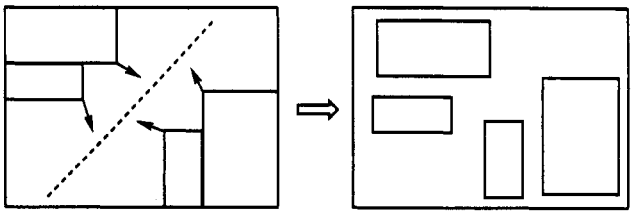

Figure 7. Relaxation of boxes toward the unified tangent.

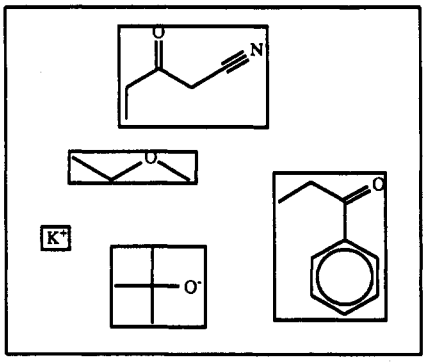

Figure 8. Effect of the analytic repositioning algorithm on the distribution in Figure 2.

where $\bar{S}_V$ is the vector in the direction normal to the Unified Tangent, with magnitude S_V:

$$\bar{S}_V = |0\ 0^{-}1| \times \bar{U}_N \cdot S_V$$

It could just as well have been approached from the H side:

$$\text{UT\_ Pnt} = P_{H,\text{red}} + (1.0 - \text{Lat\_ Scal\_ Fact}) \cdot \bar{S}_H \qquad (4)$$

C. Scaling. We now have the Unified Tangent and the three scaling factors that we sought. The largest scaling factor that will avoid overlap is the smallest of the three; we reduce that further (by 0.88) to introduce slack. If the resulting scaling factor is less than unity, scaling must occur. If it is significantly greater than unity (*ca.* 1.2), it is considered worthwhile to scale up; however, the average bond distance is not allowed to exceed a certain maximal value (*ca.* 10 character lengths).

D. Relaxation. In the final step the boxes, which are jammed together along opposite screen borders, are "relaxed" toward the empty middle. Taking the V side to illustrate (Figure 7), the leftmost V-box is shifted leftward by a fraction of the x distance to the Unified Tangent and an amount upward that centers the left edge of the box vertically between screen bottom and the unified tangent. The fraction is given by

$$\frac{n - \text{BN} + 1}{n + 1} \qquad (5)$$

where n is the number of vertical boxes (two in Figure 7) and BN is the sequence number. Thus the left V-box (BN = 1) is shifted leftward by two-thirds of the x distance from its top left corner to the tangent, while the right V-box (BN = 2) is shifted one-third of its corresponding distance. This procedure is repeated analogously for the H-boxes. Application of the algorithm to the structures in Figure 2 yields Figure 8.

III. DYNAMIC REPOSITIONING ALGORITHM

A. Introduction. The dynamic repositioning algorithm is independent from the analytic one, and may thus be used alone for special purposes (*vide supra*), but is normally applied after the other algorithm has executed. Its heart is a simple Newtonian dynamics engine. The molecules are represented

Algorithm II. Dynamic Repositioning Engine
(1) Initialize
(2) Calculate forces and tensions
  (a) Assess interactions between boxes.
  (b) Assess interactions between boxes and walls.
  (c) Calculate net forces and tensions on boxes.
(3) Accept or Reject
  (a) Decide whether to accept or reject the last movement.
  (b) Modify response (attenuation) accordingly.
  (c) If reject: restore previous coordinates; reduce shift vectors by the same amount that response was just decreased.
  (d) If accept:
    (i) Evaluate Still_Unequil and Still_Tensed flags.
    (ii) If so directed (Tension_Ratio is nonzero), scale every box's net force vector in proportion to the tension acting on it.
    (iii) Locate nearest blocking object.
    (iv) Maximize shifts, subject to the nearest blocking object.
    (v) Calculate shift vectors and apply to boxes.
  (e) Clear Still_Moving flag if no box has moved a significant amount.
  Resume at step 2 unless the maximum number of passes is exceeded or Still_Moving is clear.
(4) Postprocessing
  (a) Calculate "badness" of system: the average tension per box.
  (b) Determine if any box is inhibited.

as rectangular "boxes", surrounded by four immovable walls, representing the display borders, all of which exert repulsive forces on one another. The system is allowed to evolve until it reaches equilibrium or congestion has disappeared. Each iteration cycle has three parts. In the first the forces are evaluated and two numbers tabulated: the net force on each box, *i.e.* the sum of the individual forces acting on it, and the "tension" on each box, which is the sum of the magnitudes of the constituent force vectors. The former provides the direction in which a box will move; the latter indicates how crowded a given box is and taken collectively quantifies the extent of dispersion of the system.

In the second part the current state is compared to that in the previous iteration: if the overall tension has declined, the moves of the previous cycle are accepted; otherwise they are discarded, and the attenuation factor for moves is increased. In the third part the moves for the next iteration are calculated.

It is interesting to note that although the net force vectors derived for every box usually push the boxes apart and so decrease their overall tension, this need not always be so, and in fact it is common for the natural inclinations of the boxes to increase tension. What the force vectors always minimize (provided the movements are not too great) is the sum of the net forces. While a box may become sandwiched between two other boxes with little change in the net forces in all three (after small balancing movements), the tension of the system may nonetheless become vastly higher. It is the overall tension that we wish to minimize.

Built around the engine is a "tactics executive" that modifies the engine's parameters according to its progress.

B. Dynamic Repositioning Engine. An outline of the engine is given in Algorithm II; a detailed discussion of those steps now follows. Variable names appear in italics.

Calling arguments (supplied) include:

(a) *Minim_Goal.* This determines what quantity is being minimized, *i.e.* which variable must decrease in a cycle for that cycle's movement to be accepted. In practice this is tension, but this flag makes it possible to minimize instead the sum of net force magnitudes (true equilibration).

(b) *Completeness.* This value from 1 to 10 controls how thoroughly the dynamics proceeds. A larger value reduces the thresholds for minimal move distances and tensions and thus extends the run and evokes greater refinement.

(c) *Movable.* This is a vector of flags indicating whether each box is static or movable.

(d) *Tension_Ratio.* If positive, boxes under higher tension are made more responsive than those under lower tension. If negative, *vice versa.* If zero, tension does not affect box movement. The magnitude is the ratio of scaling between the most tensed and least tensed boxes; the scaling factors average to 1.0.

Calling arguments (returned) are:

(a) *Congestion.* An absolute measure of the system's congestion.

(b) *Progress.* True if the system was improved, however little.

(c) *BoxStuck.* True if one or more boxes are trapped in a local minimum; tips the tactics executive that a change in the calling arguments is in order.

(d) *Still_Moving.* True if progress was being made at the time the maximum iterations count was reached.

(e) *Still_Tensed.* True if the spread in tensions in the boxes is large or if the average tension per box is high; indicates that dynamics should be continued.

(f) *Still_Unequil.* True if the sum of the net force magnitudes is judged to be larger than that attainable; indicates that the system is out of equilibrium, and like *Still_Unequil*, that more improvement should be possible.

(1) Initialization. A number of factors and threshold values are set, based on the calling arguments:

(a) *Trivial_Movement*, in pixels, is set to

$$\mathrm{MAX}(1,5/\mathrm{completeness})$$

and ranges from 1 to 5. If in any cycle no box moves more than this amount, the system is considered to have halted and the engine terminates.

(b) *Response*, the amplification factor applied to shifts, is set to 1.0. When the system is evolving in a beneficial direction, as evidenced by the acceptance of the previous cycle's shift, *Response* is multiplied by 1.1; when a shift is rejected. *Response* is divided by 4.

(2) Calculate forces and tensions.

(a) Interactions between boxes. The force between two boxes is made inversely proportional to the distance (the distance-squared gradient is too short-ranged) between their two closest points of approach (eq 6). The tension

$$|\bar{F}| = 100/r \quad (6)$$

on a box is the sum of these magnitudes. To avoid infinite forces, a floor value (five pixels) is applied to the distance used. In order that interpenetrating boxes should have an impetus to separate, a second force component is added

$$\bar{F}(b_1,b_2) = 0.2(\mathrm{MMV}_{max}(b_1,b_2) - |\overline{\mathrm{MMV}}(b_1,b_2)|) \quad (7)$$

where $|\overline{\mathrm{MMV}}|$ is the distance between the box centers (the mean-to-mean vector) and MMV_{max} is the maximum value possible for the two given boxes.

(b) Interactions between boxes and the four walls. The walls are imaginary constructs coincident with the borders of the display; they are represented as one-pixel thick static boxes. The force and tension between each wall and a box is figured as in the previous step. Additional static objects may be present, as determined by the *Movable* calling argument.

(c) Sum forces

(i) *NetFMag(box#)* is the net force acting on a given box.

(ii) *SumNetFMag* is the sum of the net force magnitudes over all movable boxes. It measures how out of equilibrium the system is.

(iii) The minimum, maximum, and sum of all boxes' tensions are compiled.

(3) Accept or reject the previous cycle's shifts.

(a) Since the initial state does not result from a trial shift, it is accepted automatically. If a specified number of cycles has been exceeded, the state is rejected. Otherwise, the state is accepted if the quantity being minimized (normally the total tension) is smaller than in the previous cycle.

(b) *Response* is augmented or diminished, depending on whether or not the cycle was accepted or rejected, respectively. It is not modified on the first cycle.

(c) If rejected: the previous coordinates are restored, and the shift vectors are reduced by the same factor as that just applied to *Response*. The new shift vector are added to the boxes' coordinates.

(d) If accepted:

(i) The *Progress* flag is set, and the flags *Still_Unequil* and *Still_Tensed* are evaluated. Because this step must be executed on the first pass, these flags are certain to be defined on exit. The system is considered unequilibrated if the average net force per box is more than $10/completeness$. The system is considered tensed if the largest tension is more than three times the smallest (since ideally tension would be distributed evenly) or if the smallest tesnion divided by the number of boxes is larger than some constant.

(ii) If the calling argument *Tension_Ratio* is nonzero, and if there is a significant ($\mathrm{MAX}/\overline{\mathrm{MIN}} > 1.5$) spread in box tensions, the net force acting on each box is increased or decreased proportionately according to the box's tension (see below).

(iii) The purpose of this step is to determine the shift that would advance the system as far as possible without overlapping walls or other boxes. The first blocking object is located that would be encountered if the boxes were moved along and in proportion to their net force vectors. Possible blocking objects are the four walls and the other boxes, including static ones. For every movable box, its shilhouette is cast in its direction of motion, forming a skewed rectangular shape. The intersection of this shape with every other object is calculated. As can be seen from Figure 9, the blocking distance is somewhat inaccurate due to the uncertainty about the exact points of contact. This is not very serious, however, in part because the distance is underestimated rather than overestimated.

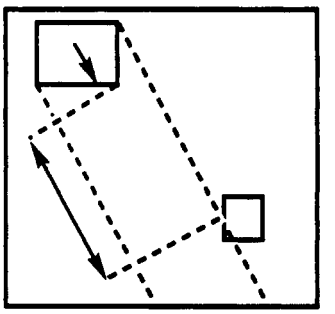

Figure 9. Blocking distance model may underestimate blocking distance slightly.

To establish common units of susceptibility, since each box will be moved in proportion to the net force acting on it, each box (B)'s blocking distance is divided by this magnitude:

$$\mathrm{Move\_Limit}(B) = \frac{\mathrm{Block\ Dist}(B)}{|\bar{F}(B)|} \quad (8)$$

(Conceptually, the units of *Move_Limit* are time: how long a given box can travel at its "velocity" before it collides.) The box that is most limited in movement is that with the smallest value, $Move\_Limit_{min}$.

(iv) The shift vector of a box B will be $\bar{S}(B)$:

$$\mathrm{Move\_Factor} = \frac{\mathrm{Response} \times \mathrm{Move\ Limit}_{min}}{\mathrm{Damper}^{-}} \quad (9)$$

$$\bar{S}(B) = \mathrm{Move\_Factor} \times \bar{F}(B) \quad (10)$$

where *Damper* is 3.0. The distance boxes move is therefore composed of the following effects: (1) The maximum any box B can travel without one of them running into another object is $Move\_Limit_{min} \times \bar{F}(B)$; (2) to be certain no collision occurs, an attenuation (*Damper*) is applied; and (3) modification by previous experience (*Response*) reflects the temperarature of the system.[5]

This formulation has the disadvantage of being dependent on the vagaries of changing blocking distances. If a box's force vector changes direction slightly, for example, its blocking distance could change dramatically while its optimal shift probably would not be very different. It might be advantageous to average *Move_Factor* with that of the previous cycle.

The largest move of any box is not allowed to exceed about $^1/_{20}$th of the display width. This prevents violent changes in the pattern of boxes and hence of the force field.

(v) The shift vectors are obtained by eq 10 and added to the boxes' coordinates.

(e) If no box has moved a significant amount in the shift just applied, the *Still_*Moving flag is cleared. This test applies to the reject condition as well, because in step 3c an appropriate shift was calculated.

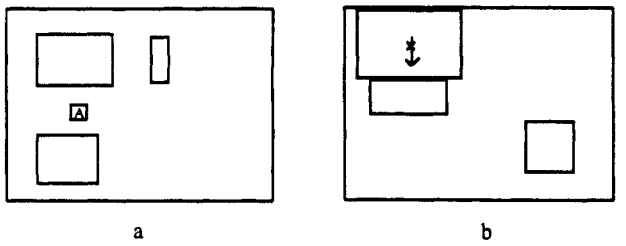

Figure 10. Inhibition of dynamic repositioning: (a) frozen by tension, (molecule A is in such a deep potential well that even small movements are impossible). (b) stymied by jamming (the top box is blocked by the lower, and so neither moves).

Looping continues until the maximum number of passes is exceeded or the boxes are not moving significantly (*Still_Moving* is clear).

(4) The congestion of the system is calculated as the average tension per box. Also, the boxes are examined to determine if any are inhibited, as evidenced by having large net force vectors yet not moving. A box is considered stuck if the net force acting on it is at least 20% of its tension.

C. Tactics Executive. The engine just described is not always effective. The most common problem is that one box is under such high tension that it cannot move (it is near, but not at the bottom of, a steep potential well), yet other boxes are prevented from moving because the nature of the cycle is to accept or reject the collective motion of all boxes (Figure 10a). This phenomenon is more prevalent when *Tension_Ratio* is positive, meaning that the jumps of high-tension-boxes are amplified. Similarly, though not ordinarily encountered, a box can be so tightly jammed in place that its blocking distance would be zero, again arresting the other boxes along with itself (Figure 10b). For this reason the engine is embedded in an executive routine that changes tactics when the boxes stop moving but are still too tensed. There are three modes, applied one after the other and repeated until a maximum number of iterations is reached or the system is perceived as optimized:

(1) Normal

(2) Inverse Tension. The effect of a box's tension on its movement is reversed. Whereas normally the shifts of boxes under higher tension are amplified relative to those under lower tension, which usually speeds convergence by concentrating on the crucial boxes, an inverted tension ratio causes boxes under less tension to be more agile. This often frees up the boxes that have become stuck for either of the reasons just mentioned.

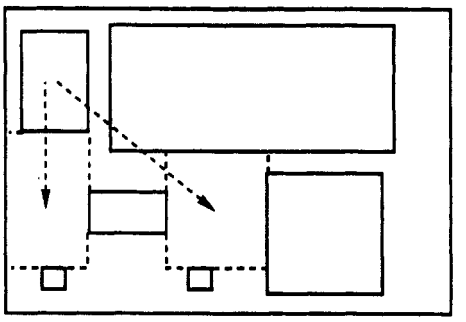

Figure 11. Possible jumps in the jumping heuristic. The borders of the free rectangles involved are indicated with dashed lines. Both jumps would improve the system after equilibration.

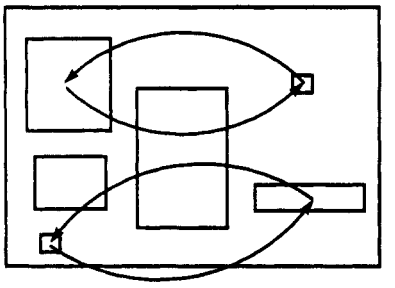

Figure 12. Possible swaps in the jumping heuristic. Both of the exchanges illustrated would improve the system after equilibration.

(3) Partial Box Mode. A second way of loosening a locked configuration is to focus on one box at a time. All other boxes are marked as static, and the special box is allowed to minimize the tension upon itself at the expense of the other boxes. This is repeated for every box. The effects are often dramatic, as when small molecules are suddenly freed to dart from their previously locked positions. This mode is not the standard because it executes more slowly, and the relative locations of boxes are less likely to be preserved.

If these tactics fail, then either the system is very dense, and a significantly better solution does not exist, or else the system is locked into a local minimum. A "Jumping Heuristic" is provided that scrambles boxes intelligently; it is explored next. The overall dynamic repositioning strategy is summarized in Algorithm III.

D. Jumping Heuristic. The Jumping Heuristic is used when the system is not developing but the tension per box is still high. The principle is to extricate a box from a crowded region and start it in a different, uncrowded portion of the screen. There are two ways this is done: by moving a crowded box to a large empty area (the "jump", Figure 11) and by swapping such a box with a smaller box in an uncrowded area (the

Algorithm III. Dynamic Repositioning Strategy

Calling Arguments (supplied) include the following.

(a) *Atoms*: The subset of the connection table to be repositioned. Unlike the analytical repositioning algorithm, the dynamic algorithm can reposition one or more molecules among other static molecules.

(b) *Static Boxes*: A list of the molecules not in *Atoms*, plus any imaginary obstructions. The latter are used to steer molecules clear of areas of the screen that will be occupied by special graphics. An entry in the list consists of the four coordinates needed to describe a box.

(1) Perception and Initialization

(a) Create boxes to represent molecules.

(b) The static boxes, if any, are appended to the list of ordinary boxes. The "Movable" flag for each is permanently cleared. The others' flags are set.

(c) The iterations counter is cleared.

(d) Normal tension ratios are put into effect.

(2) The Tactics Executive is invoked. *Completeness* is provided at a low value for a quick and crude treatment. Thus a search for other global minima is made before the engine refines a state that is not very good.

(3) If the system is stuck, the Jumping Heuristic is applied; if it is successful, control is returned to step 2.

(4) The Tactics Executive is invoked with a high value of *Completeness* in order to develop the system most thoroughly.

(5) Coordinate changes are written back to the connection table.

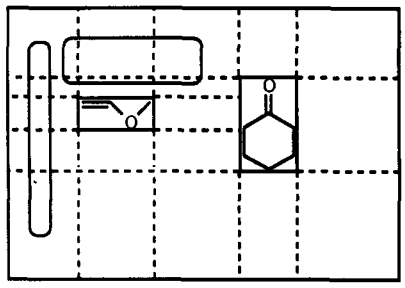

Figure 13. Free rectangles (FR's) in a simple system. An FR is the largest rectangle drawable without including molecule boxes. The dashed lines are FR borders; note that FR's overlap heavily. The positions of two FR's are indicated by the rounded rectangles inside them.

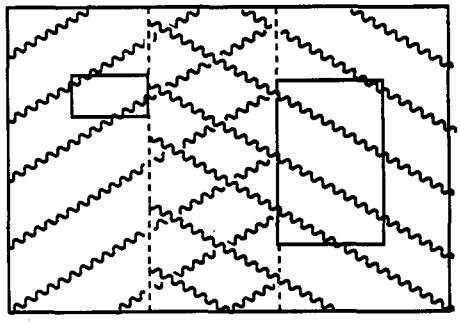

Figure 14. Two superboxes (shaded regions) corresponding to two molecule boxes.

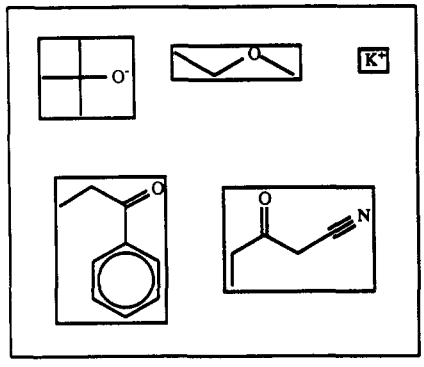

Figure 15. Result of dynamic repositioning applied to the structures in Figure 2. The ether was "jumped" to the top.

"swap", Figure 12). The procedure is outlined in Algorithm IV.

1. **Initialization.** In the initialization part every box is assigned a "move rating" (M) that estimates the benefits of moving that box away from its present position. M is composed of two terms (eq 11): one that includes the size of the box, because moving large boxes reshuffles the system more than moving small ones, and one that includes the box's tension, because moving high-tension boxes is the best way to reduce overall tension. Very large boxes are discarded because it is unlikely that moving these dominant objects would improve the system. The worst 50% are also discarded, except that at least two are kept if possible. The resulting boxes are called the "Most Movable Boxes", or MMB's.

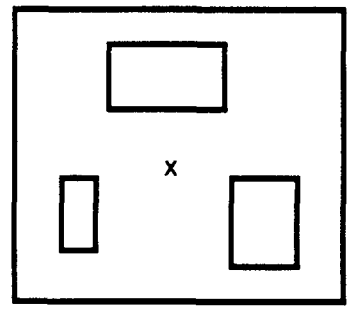

Figure 16. Unified repositioning algorithm. The next molecule will be placed in the largest screen space ("X").

$$M(\text{box}) = \frac{1}{5}\left[\frac{\text{area(box)}}{\text{area}_{\max}}\right]^{1/2} + \left[\frac{\text{tension(box)}}{T_{\max}}\right]^{1/2} \quad (11)$$

$\text{area}_{\max}$ = area of the largest box

$T_{\max}$ = tension on the most tensed box

Also, a list of the unoccupied portions of the screen, the "free rectangles" (FR's), is compiled. These will be used not only for the jumping tactic but also in assessing the merits of a particular swap. Figure 13illustrates the free rectangles in a simple system.

2. **Jumping.** Jumping is attempted before swapping because it is usually more effective, as well as being computationally simpler. Each MMB is paired with every free rectangle 90% of its size or larger, and a match rating (R_J) is calculated (eqs 12 and 13). Note that YFit is calculated analogously to XFit

$$R_J(\text{box,FR}) = (\text{XFit} + \text{YFit})\left[\frac{\text{area(FR)}}{\text{area}_{\max}}\right]^{1/2} + 0.5M(\text{box}) \quad (12)$$

in eq 13. The rating is a function of whether the box fits easily into the FR (represented by XFit + YFit), how the FR

Algorithm IV. Jumping Heuristic

(I) Initialization
- (1) To every box assign a movability rating reflecting the strategic value of dislodging the box from its current position. Select the best of these as the "most movable boxes" (MMB's).
- (2) Compile a list of the free screen areas in the form of (overlapping) rectangles. These are the "Free Rectangles" (FR's).

(II) Jumping: a box is moved to a free screen space
- (1) Rate every pairing of an MMB with an FR big enough to receive it. Discard all but the most promising.
- (2) For every pairing, make the jump and invoke the dynamics engine to equilibrate the new arrangement crudely. Save improvements; if any improvement makes the system congestion-free, the new state is kept and the procedure exits directly with success status.

(III) Swapping: a large box under high tension is exchanged for a smaller box with much empty space around it
- (1) The "superbox" is calculated for every box: the amount of area including and around the box that would be free were the box to be moved.
- (2) MMB's are paired with smaller boxes that have large superboxes, and a match rating is calculated for each such pair.
- (3) Pairings between boxes of similar size are discarded.
- (4) As in step II.2, the effects of each swap are investigated by cursory dynamics. If any improvement makes the system congestion-free, the new state is kept and the procedure exits directly with success status.

(IV) If any of the experiments decreased the overall system tension, the best of these is adopted; in any case the heuristic exits with failure status, meaning that the system is still congested.

Algorithm V. Maximal Lateral Scaling

(1) Categorize molecules as V-boxes or H-boxes, and calculate XScale and YScale.
(2) Calculate $\bar{C}_V$ and $\bar{C}_H$ for all boxes by eq 1.
(3) Lat_Scale = 10^{10}
(4) (a) Loop over $v \in$ V-boxes. Let V_0 be the box's top left corner.
(b) Loop over $h \in$ H-boxes. Let H_0 be the box's bottom right corner.
(i) Let $\bar{C}_{tot.} = \bar{C}_V(v) - \bar{C}_H(h)$ represent the relative motion of v relative to h as the system is scaled.
(ii) Find t_1 such that

$$V_0(x) + t_1\bar{C}_{tot.}(x) = H_0(x)$$

and t_2 such that

$$V_0(y) + t_2\bar{C}_{tot.}(y) = H_0(y)$$

(iii) Boxes v and h will collide at a scale factor $t_{max} = \mathrm{MAX}(t_1,t_2)$.
(iv) Lat_Scale = MIN(Lat_Scale,t_{max}).
(5) The maximum scaling factor that falls just short of overlap is MIN(XScale,YScale,Lat_Scale). Scale by a somewhat smaller amount.
(6) Invoke dynamic positioning to space boxes apart.

Algorithm VI. Unified Repositioning Algorithm

(1) Rank molecules by area.
(2) Place the largest three in the pattern of a triangle. For simplicity, molecular size need not be considered.
(3) Dynamically reposition the system. If two boxes are overlapping or too close, scale the system down and restart this step.
(4) If there are boxes remaining, place the largest in the largest available screen space and resume at step 3.
(5) (optional) If the smallest distance between two boxes is rather great and if the molecules are not already rather large, scale the system up and go to step 3.

$$\mathrm{XFit} = \frac{\text{FR's } X \text{ extent}}{\text{Box's } X \text{ extent}}; \quad \text{if (XFit} > 1.0\text{)},$$

$$\text{then XFit} = 1.0 + \frac{(\mathrm{XFit} - 1.0)}{4} \quad (13)$$

compares in size with other FR's, and how strategic it is to move the particular box. The worst 70% are discarded, and the remaining, examined. For each, the tactics executive is invoked in its cursory mode. If an improvement occurs that is so good that the system becomes acceptable, as evidenced by the *Still_Unequil* flag returning clear, the Jumping Heuristic terminates. If the improvement is only meager, the improved coordinates are noted but not adopted until the end of the procedure.

3. Swapping. If jumping does not disentangle the boxes, swapping is tried. In this mode a large box under high tension is exchanged for a smaller box that is surrounded by a generous empty buffer. This arrangement is usually beneficial because the tension released by the large box will be greater than that gained by the small box. To this end, first the "superbox" surrounding every box is calculated. This is the largest free rectangular region surrounding and including the box itself (Figure 14); it represents the effective area that is available to another box swapping in. Next, each MMB is paired with every other box (OB) that is smaller and whose superbox is at least 80% as large as the MMB. A match rating (R_S) is calculated as follows:

$$R_S = \mathrm{Ease(MMB)} + \mathrm{Ease(OB)} + 0.5\,M(\mathrm{MMB}) + 0.5\left[\frac{\text{tension(MMB)}}{\text{tension(OB)}}\right]^{1/2} \quad (14)$$

where

$$\mathrm{Ease(B)} = \mathrm{MIN(XEase(B), YEase(B))}$$

$$\mathrm{XEase(B)} = \frac{X \text{ extent of } B^C \text{ superbox}}{X \text{ extent of } B}$$

B^C is whichever of OB and MMB that is not B, *i.e.* MMB + OB − B

and other variables are defined analogously. *Ease(MMB)* estimates how well MMB can snuggle into OB's spot, and *Ease(OB)* measures how well OB can fit into MMB's old spot. The term involving tension favors moves which bring a box under high tension (MMB) to a spot of low tension (OB).

Only the best six pairings are accumulated. In order to further weed out unproductive combinations, each selected pair is reanalyzed. If the difference in X (and separately in Y) extents between the MMB and OB is less than 10% of the free extent around the OB, the pairing is discarded, because the boxes are too close in size for a swap to have much effect on the system. Finally, only pairings with ratings within 30% of the best value are retained.

The surviving pairings are then explored one at a time, as with the jump. A swap is executed, cursory dynamics are carried out, and any improvement is noted. If an improvement is such that the system is no longer very tensed, the Jumping Heuristic terminates. After all pairings have been examined, the best one (if there are any) of all the jumps and swaps is retained, but the Jumping Heuristic returns failure status nonetheless because the system is still too tense.

Figure 8 is an example where the product of the analytic algorithm cannot be improved by dynamic repositioning, which leaves the arrangement unchanged. For illustration, dynamic repositioning applied directly to the precursor, Figure 2, would yield the configuration in Figure 15.

IV. ALTERNATIVE REPOSITIONING STRATEGIES

Although the analytic repositioning algorithm is effective at packing molecules together so that they may remain as large as possible, it is longer and more complicated than intuition suggests it should be. Furthermore it still, on occasion, reduces molecular size more than is necessary. One wonders if there is a simpler way and if the analytic and dynamic algorithms could somehow be combined into one. Two possible answers follow.

A. Modified Analytic Repositioning Algorithm. The modified analytic repositioning algorithm begins the same way as the original: boxes are arranged into V and H classes, and XScale and YScale are calculated as before. The lateral scaling factor is maximized, at the cost of losing the straight Unified Tangent separating line. This makes the simple relaxation step impossible, but instead the system is entrusted directly to dynamics immediately after scaling. Calculation of the maximal lateral scaling factor is presented in Algorithm

V. (It can be speeded up by noting that not all combinations of v and h need be examined.) A drawback to this version is that the analytic algorithms has become dependent on the dynamic one.

B. Unified Repositioning Algorithm. An alternative solution would be to abandon the analytic procedure altogether and empower the dynamic algorithm segment to scale as necessary (Algorithm VI). Molecules from whatever source are fed one at a time to the emptiest spots on the screen (Figure 16), the resulting system being dynamically repositioned and rescaled as necessary. The simplicity of this approach is attractive, but it might be unacceptably slow for real-time applications on small computers, and it would probably not maintain large molecular size as effectively as the analytic procedure.

C. Other Refinements. As most molecules do not fill the bounding rectangles, the dynamic repositioning alogorithm would model molecules better as aggregates of convex irregular polygons or smaller rectangles rather than as discrete large rectangles (cf. the nitrile in Figure 8). Adoption of a more detailed representation would slow down the operation. It would also increase the incidence of false local minima, although the jumping heuristic should combat these effectively.

V. CONCLUSIONS

A method of distributing a collection of molecules evenly about the computer screen has been presented. It consists of two complementary, independent procedures, one specializing in determining good relative positions, the other in evening the spacing between molecules. The first is analytical and rapid, while the other is a slower iterative minimization procedure. Both are sufficiently rapid, however, to provide pleasing results in a small fraction of a second, even on a microcomputer.

ACKNOWLEDGMENT

Gratitude is expressed to the National Science Foundation for support of this work.

REFERENCES AND NOTES

(1) For a recent review of the CAMEO program, see: Jorgensen, W. L.; Laird, E. R.; Gushurst, A. J.; Fleischer, J. M.; Gothe, S. A.; Helson, H. E.; Paderes, G. D.; Sinclair, S. CAMEO: a program for the logical prediction of the products of organic reactions. *Pure Appl. Chem.* **1990**, *62*, 1921–1932.

(2) The DEPICT program employs a simple trick to guarantee that molecules do not overlap after their bond lengths and angles have been optimized. Every molecule is linked to one other by a temporary, virtual bond. After optimization the virtual bonds are excised, leading to nonoverlapping but closely positioned molecules. See: Weininger, D. Smiles. 3. Depict. Graphical Depiction of Chemical Structures. *J. Chem. Inf. Comput. Sci.*, **1990**, *30*, 237–243.

(3) The repositioning problem would appear to fall in the NP computability class, implying that there is no general solution that is simple and optimal. For a nontechnical exposition of NP (nondeterministic polynomial-time) theory, see: Harrel, D. *Algorithmics: The Spirit of Computing*; Addison-Wesley: New York, 1987.

(4) The margins added around molecules are excluded from *VSum* and *HSum*.

(5) *Damper* could be eliminated by including its effect in *Response*, but maintaining its independence reminds us of its function.