Complete Maps of Molecular-Loop Conformational Spaces

JOSEP M. PORTA,[1] LLUÍS ROS,[1] FEDERICO THOMAS,[1] FRANCESC CORCHO,[2] JOSEP CANTÓ,[2] JUAN JESÚS PÉREZ[2]
[1]*Institut de Robòtica i Informàtica Industrial (UPC-CSIC) Llorens Artigas 4-6, 08028 Barcelona, Spain*
[2]*Department of Chemical Engineering, Technical University of Catalonia (UPC), Diagonal 647, 08028 Barcelona, Spain*

Received 7 July 2006; Revised 11 January 2007; Accepted 19 February 2007
DOI 10.1002/jcc.20733
Published online 20 April 2007 in Wiley InterScience (www.interscience.wiley.com).

Abstract: This paper presents a numerical method to compute all possible conformations of distance-constrained molecular loops, i.e., loops where some interatomic distances are held fixed, while others can vary. The method is general (it can be applied to single or multiple intermingled loops of arbitrary topology) and complete (it isolates all solutions, even if they form positive-dimensional sets). Generality is achieved by reducing the problem to finding all embeddings of a set of points constrained by pairwise distances, which can be formulated as computing the roots of a system of Cayley–Menger determinants. Completeness is achieved by expressing these determinants in Bernstein form and using a numerical algorithm that exploits such form to bound all root locations at any desired precision. The method is readily parallelizable, and the current implementation can be run on single- or multiprocessor machines. Experiments are included that show the method's performance on rigid loops, mobile loops, and multiloop molecules. In all cases, complete maps including all possible conformations are obtained, thus allowing an exhaustive analysis and visualization of all pseudo-rotation paths between different conformations satisfying loop closure.

 J Comput Chem 28: 2170–2189, 2007

Key words: loop closure; ring; conformational space; complete method; Cayley–Menger determinant; Bézier clipping; tripeptide loop; disulfide bond; seven-atom loop; cycloheptane; cyclooctane; bicyclohexane; adamantane; bound smoothing

Introduction

A problem frequently arising in Molecular Modeling is the computation of the valid conformations of a molecular *loop,* a sequence of pairwise-bonded atoms, either cyclic or with fixed end-points. In this work, a *valid conformation* will be defined in a kinematic sense: as an assignment of positions to all atoms, respecting the constraints imposed by all bonds, regardless of whether the assignment provides an energetically-favorable arrangement or not. This is usually known as the molecular *loop closure* problem,[1,2] instances of which occur when determining the possible conformations of cyclic molecular structures, when looking for localized moves of an atom chain with fixed ends, or when predicting the spatial shape of nonmatched segments in protein homology modeling, for example.

When modeling a molecule, one can treat the bond-stretching and bond-bending degrees of freedom as variable quantities, subject to appropriate bond-stretching and bond-bending potentials. This realistically models the flexible nature of molecules, but it usually requires a large number of variables and expensive computations. A simpler model can be used, however, if only the principal motions of the molecule need to be elucidated. Since the stiffness constants associated with the deformation of torsion angles are one or two orders of magnitude smaller than those associated with deforming bond lengths and bond angles, it seems plausible to assume that all bond lengths and bond angles are fixed, and only torsion angles vary. This assumption, known as the *rigid-geometry* hypothesis,[3] will be adopted throughout the paper.

A rigid-geometry loop can be viewed as a closed sequence of rigid *bodies,* pairwise articulated through *hinge* joints. In simple situations the bodies and hinges are easily identified: if all atoms are allowed to rotate about their covalent bonds, we may con-

Correspondence to: L. Ros; e-mail: llros@iri.upc.edu

Contract/sponsor: The Spanish Ministry of Education and Science; contract/number: DPI2004-07358

Contract/sponsor: The "Comunitat de Treball dels Pirineus"; contract/number: 2006ITT-10004

Contract/sponsor: The Spanish Ministry of Education and Science; grant programs: Ramon y Cajal 2002/2004, 13 2006

sider every atom as a body and every torsional degree of freedom as a hinge (Figs. 1a and 1c). This yields a loop of specific geometry where all hinge axes of a same body are coincident at its atom center. More general and difficult-to-solve loops can arise, though, when one knows the relative positions of all atoms on some subchains of a long loop. In such cases, the bodies are composed of these subchains and the hinges are the rotations about the bonds that join them, yielding a body-and-hinge model with all hinge axes in general position (Figs. 1b and 1d).

Clearly, for a loop of s solids and s hinges one cannot vary all hinge angles arbitrarily without breaking the loop. Mathematically, this translates into the fact that the closure condition, in general, imposes six constraints among the s hinge angles and, thus, only $s - 6$ angles can be arbitrarily chosen, the remaining six being dependent on these.[1] Actually, the s-tuples of valid angle assignments must satisfy a system of polynomial loop closure equations, and thus from an $(s - 6)$-dimensional algebraic variety. This means that, except for degenerate cases,

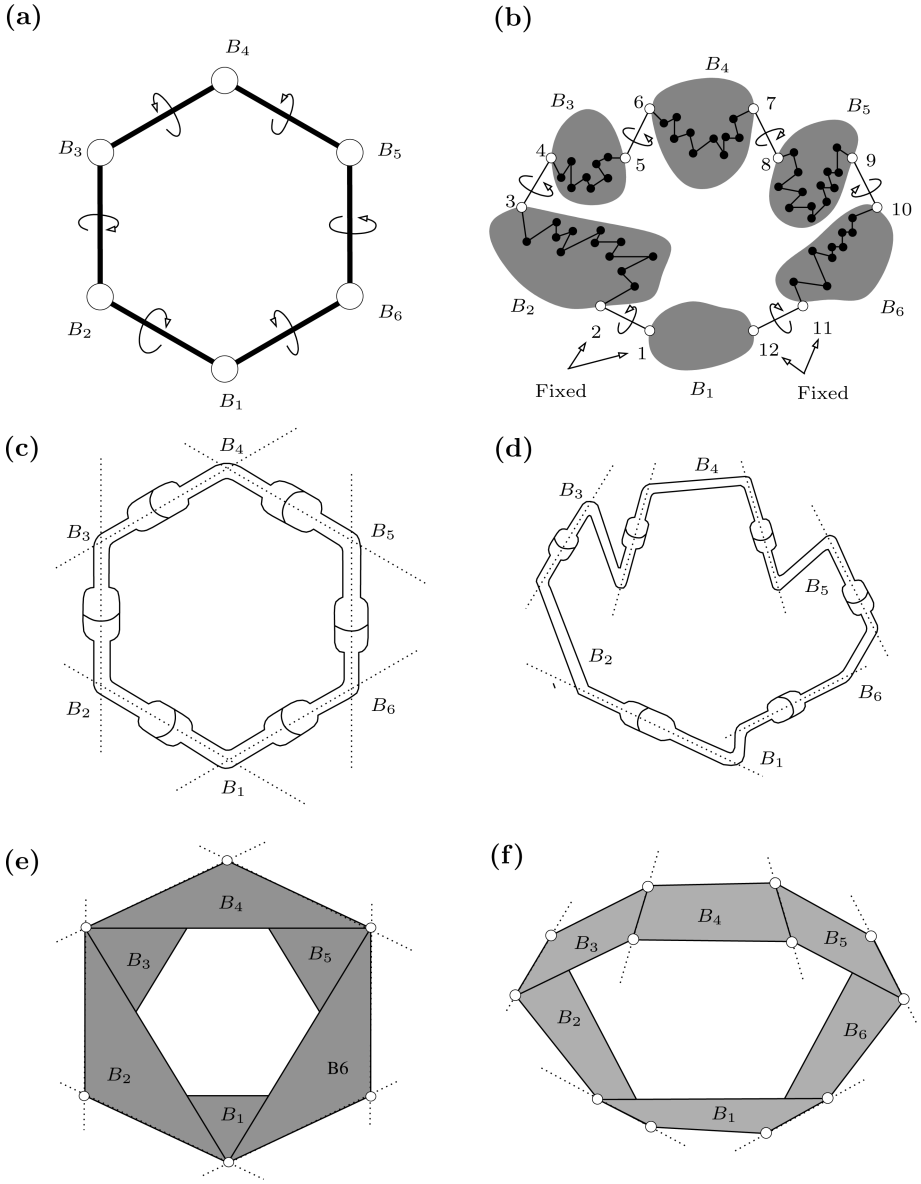

Figure 1. (a) A six-atom ring. (b) A general six-torsion loop on a long molecular chain with fixed end-points. Within the shaded subchains, the relative positions of all atoms are known and they can hence be viewed as a single rigid body. Having fixed positions in 3-space, atoms 1, 2, 11, and 12 also define a rigid body. Free torsion angles are indicated with circular arrows. (c) and (d) Body-and-hinge models of the two loops, with the bodies labelled $B_1, \ldots, B_6$. (e) and (f) Their distance models.

1. If $s < 6$ the loop is overconstrained and cannot be assembled. (The variety is the empty set.)
2. If $s = 6$ the loop is well-constrained, but it can only adopt a finite number of *rigid* conformations.* (The variety is a finite collection of isolated points.)
3. If $s > 6$ the loop is underconstrained and it can adopt infinitely many *mobile* conformations along continuous pseudo-rotation paths. (The variety is a finite number of positive-dimensional connected components.)

The solution of loop closure equations is in general difficult, especially when the loops are long (with $s > 6$) and all possible solutions are required. Thus, a common trend in the literature has been to provide exact methods for the $s = 6$ case, and use them to explore the conformational space of longer loops by repeatedly sampling $s - 6$ angles. Using elimination techniques and exploiting geometric particularities such as the planarity of peptide bonds or the coincidence of consecutive hinge axes, for example, efficient *ad hoc* solutions have been given for the six-atom ring, the seven-atom chain, the tripeptide loop, and the disulfide bond, to name a few,[1,2,4,5] but no efficient technique for general multiloop structures has been given yet.

Rather than following such trend, this paper presents a method to solve the problem in its full generality. The method is able to find all spatial conformations of single or multiple intermingled loops, irrespectively of the loop lengths, of the relative positions of the hinge axes, and of the loops' interconnection pattern. In all cases, the method returns complete approximations of the conformational space, given as collections of small boxes that contain all of its points. These approximations, moreover, can be refined to any desired precision. The result is a discrete map of the whole conformational space where isolated boxes correspond to rigid conformations and sets of adjacent boxes correspond to positive-dimensional components associated with pseudo-rotation paths satisfying loop closure.

The rest of the paper is organized as follows. Section Distance Models prepares the ground and shows how the problem can be formulated as one of satisfying a number of distance constraints between points. Based on such formulation, Section Related Work presents a unified view of previous solution methods from Molecular Modeling and Robotics. Section Computational Method develops the proposed loop-closure strategy: It first shows how the problem reduces to finding the zero-set of a system of Cayley–Menger determinants, then presents a technique to solve such systems, and finally analyzes its computational cost. Section Study Cases illustrates the method's performance on various single- and multiloop structures: the tripeptide loop, the seven-atom ring, the disulfide bond, and adamantane (which are rigid) and cyclohexane, cycloheptane, cyclooctane and bicyclohexane (which are mobile). The paper's conclusions and points deserving further attention are finally summarized in the end.

Distance Models

Mathematically, molecular loops can be modeled as systems of distance constraints between points.[6] Assuming the rigid-geometry hypothesis, such modeling entails placing a point for each atom, and fixing enough distances between the atoms to lock all bond lengths, bond angles, and torsion angles that are constant in the molecule. A constant bond length simply implies a fixed distance between two atoms. A constant bond angle θ between atoms *i, j,* and *k* is specified by fixing the $i - k$ distance (Fig. 2a). A constant torsion angle ϕ on four consecutive atoms *h, i, j, k,* can be set by specifying the $h - k$ distance (Fig. 2b). Finally, if the torsion angles within *m* consecutive atoms $a_1, \ldots, a_m$ are constant, the *m* atoms can be seen as a rigid body, which can be modeled by fixing the distances of the tetrahedron defined by a_1, a_2, a_{m-1}, a_m (Fig. 2c).

The points and distances defined in this way will be hereafter referred to as the *distance model* of the loop. Figures 1e and 1f provide distance models for the six-atom ring and the general six-torsion loop. Observe that, in essence, we shall always obtain a ring of hinged triangles or tetrahedra, where each triangle (tetrahedron) models a rigid body between two intersecting (skew) hinge axes. For ease of visualization; crosslinking bars of the tetrahedra involved in such models will be omitted in Figures 1 and 3.

In sum, a molecular loop can be represented as a graph $G = (V, E)$, and an assignment $d : E \to \mathbb{R}$, where each vertex in V represents one of ν chosen points $\mathbf{p}_1, \ldots, \mathbf{p}_\nu$, and each edge (i,j) in E represents a pair of points $\mathbf{p}_i$ and $\mathbf{p}_j$ whose distance is fixed to the value $d(i,j)$. With these definitions, the molecular loop

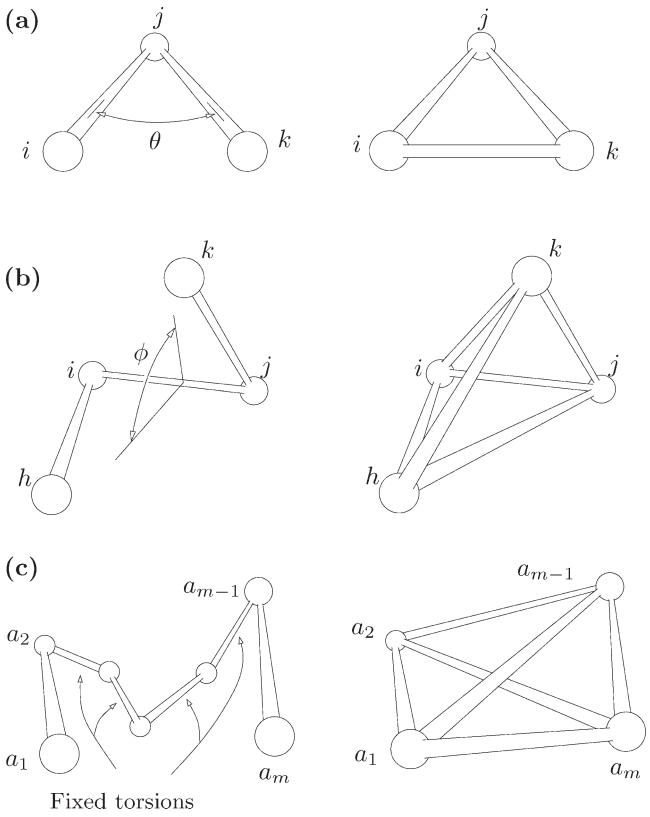

Figure 2. Translation of bond-bending (a), torsion-angle (b), and rigid-body (c) constraints. Bars between atoms indicate fixed distances.

*Throughout the paper a rigid conformation is one that cannot be deformed while keeping all loop closure constraints satisfied. A conformation is said to be *mobile* if such deformation can be done.

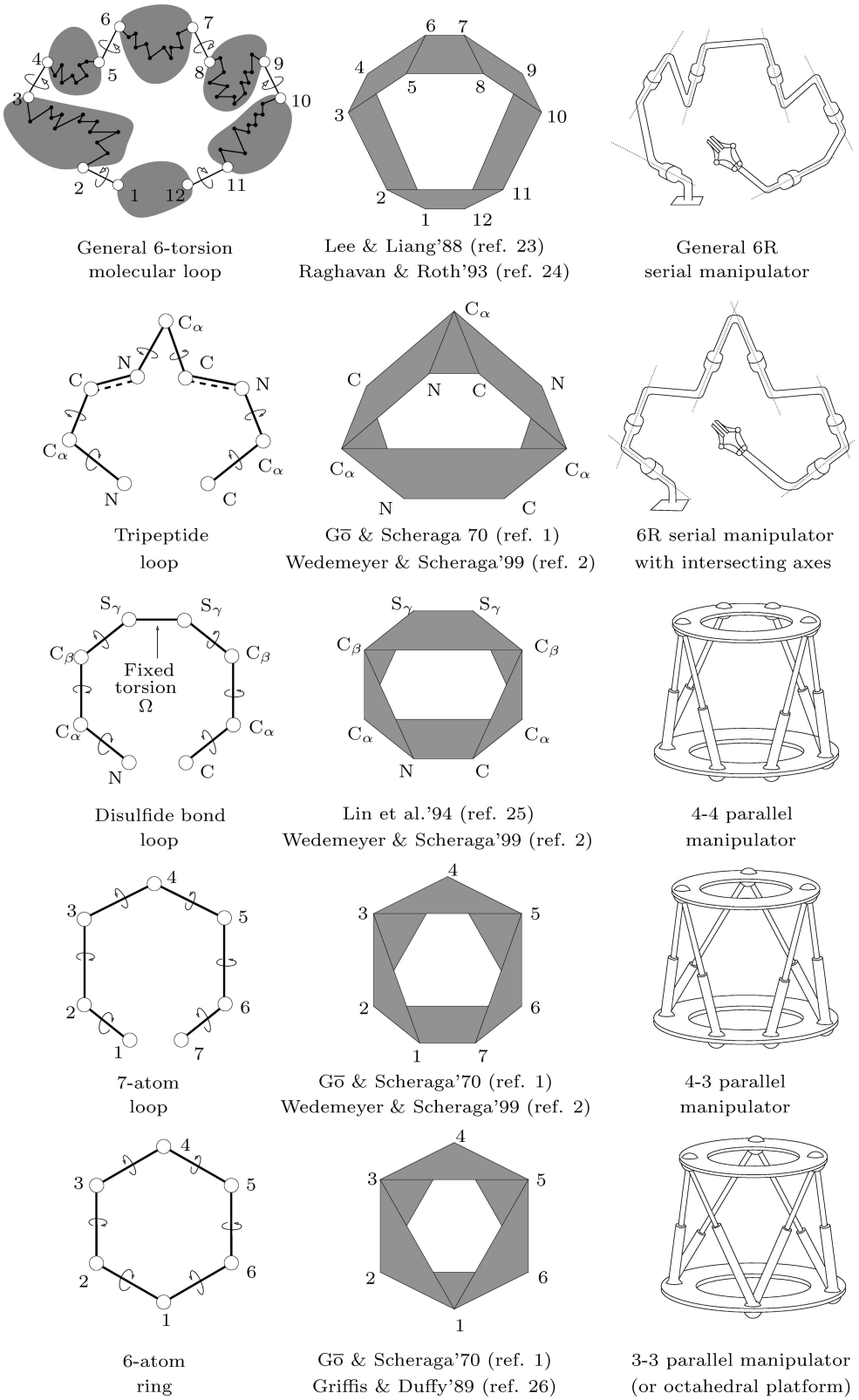

Figure 3. The mainly studied molecular loops (left) together with their distance models (center) and equivalent robots (right). For each loop, we cite the main applicable techniques under its distance model. All loops can attain, at most, 16 conformations.

closure problem is simply to determine whether there exist actual vectors $\mathbf{p}_1, \ldots, \mathbf{p}_v \in \mathbb{R}^3$ such that

$$d(i,j) = \|\mathbf{p}_i - \mathbf{p}_j\|, \quad \text{for all } (i,j) \in E,$$

where $\|x\|$ denotes the Euclidean norm of $\mathbf{x} \in \mathbb{R}^3$. If they exist, these vectors are said to form a feasible *embedding* of G and our interest is to map out (up to isometries) the whole set of such embeddings, i.e., the entire conformational space of the molecular loop. Unfortunately, although some problem subclasses can be solved efficiently,[7,8] the problem is NP-hard in general.[9] Thus, the best we can expect in our case is to come up with an algorithm that works well for problems of reasonable size, despite the unavoidable exponential complexity.

Viewing a loop as a distance model presents three main advantages. First, NMR-derived ranges for some interatomic distances can be readily used to reduce the size of the search space: they simply impose upper and lower bounds on some of the involved distances. Second, the conditions that the points must fulfill to be embeddable in 3-space are well-characterized.[10] They encompass a number of determinantal equalities and inequalities involving point-to-point distances only (see Subsection Loop Closure Equations). Following refs. 6 and 11, this permits formulating the loop closure problem directly as a polynomial system of equations, which avoids the algebraization step required by other approaches departing from trigonometric relations among torsion angles.[1,2,5] Third, a distance model is convenient to predict the *mobility* of the molecule, i.e., the dimension of its conformational space. Although, this is an easy task on a single loop, it is not trivial at all on complex molecules with multiple interconnected loops, where regions with zero, one, or larger mobility might coexist. Detecting such regions is crucial not only to predict which atoms can undergo finite motions, but also to develop divide-and-conquer strategies to tackle the loop closure problem in a modular way. A wealth of results from Rigidity Theory can be used to this end,[12] which are already available as implemented algorithms in recent packages such as FIRST[13–15] or ProFlex.[16]

Related Work

Efficient solutions to the molecular loop closure problem have been pursued for over three decades within Molecular Modeling. Already in the sixties, the pioneering work by Gō and Scheraga pinpointed the soundness of rigid-geometry approaches to low-energy conformational search,[17] and provided the first solutions for several loop topologies.[1] Although posed in a different vocabulary, the issue has also surfaced in modern Robotics, where it has been a topic of major research during the same time span, specially after the early work by Pieper[18] and Roth et al.[19] on the position analysis of robot manipulators. Given the interdisciplinarity of the problem, we opt for surveying relevant techniques from both fields. We shall focus on exact, complete methods for rigid loops (with $s = 6$). Sampling methods for longer loops, or local search methods based on minimizing an objective function also abound, but the former usually rely on an exact method for the $s = 6$ case,[2] and surveys of the latter can be found in refs. 20–22.

The kinematic equivalence between molecular and robotic loops can be better appreciated by resorting to their distance models. Figure 3 gives such models for the most-studied loop topologies: the general six-torsion loop, the tripeptide loop, the disulfide bond, the generic seven-atom loop, and the six-atom ring. According to ref. 27, the robots in the right column have the same distance models, meaning that the loops in the left column correspond either to a serial 6R manipulator (an arm with six hinged bodies, and a gripper mounted on one end) or to a parallel one (a platform linked to a base through six actuated leg-pistons). Assuming that the gripper's position relative to the ground is known in the former case, and that the leg lengths are fixed in the latter, the goal in Robotics is to find all configurations of the robot links that respect such constraints. This is clearly analogous to finding the kinematically-feasible conformations of the related molecules.

Actually, the models also reveal that the last four loops in the figure are successive specializations of the first one. To obtain the tripeptide loop, for example, reduce edges (2, 3), (6, 7), and (10, 11) to single points on the general six-torsion loop. Reducing (2, 3), (4, 5), (8, 9), and (10, 11), contrarily, yields the disulfide bond structure. The seven-atom loop can be viewed as a special case of both the tripeptide or the disulfide bond loops: one can reduce the edge between the two S_γ atoms on the latter, for example. Finally, the six-atom ring is trivially derived by reducing (1, 7) to a single point on the seven-atom loop. It is important to realize this hierarchy of specializations, since any method we cite next for a loop, actually works for any of its special cases.

The tripeptide loop, the seven-atom loop, and the six-atom ring were first solved by Gō and Scheraga in 1970.[1] They tackled the problem by assigning a reference frame to each atom and then writing the matrix equations that allow moving from one frame to a vicinal one in the loop. Since one must obtain the identity matrix when composing all such moves, this gives a necessary and sufficient condition for loop closure. For a general loop, the condition yields a system of trigonometric equations that is difficult to solve. However, Gō and Scheraga found that it can be reduced to a univariate polynomial equation in the afore-mentioned three cases, thus permitting the computation of all loop conformations via fast polynomial root finding techniques. This result was largely unnoticed in the Robotics literature of that time, and it was not until the work by Griffis and Duffy,[26] for example, that the octahedral manipulator equivalent to the six-atom ring was considered solved.

During the 80s, a quest started within Robotics for an exact solution to the general six-torsion loop. Although homotopy techniques were also competing,[28] the preferred strategy was the algebraization of the loop equations and their reduction to a *resultant,* a univariate polynomial whose solutions, once backsubstituted into intermediate equations, permitted the efficient computation of all possible conformations. Although many topological subcases were solved, it was not until 1986 that Primrose proved the six-torsion loop could adopt, at most, 16 different conformations.[29] Since Pieper had already found an example with exactly 16 conformations,[18] this was a tight bound. Soon after, the problem was definitely

settled by Lee and Liang,[23] and Raghavan and Roth,[24] who gave the first 16th-degree resultants for this loop, and by Manocha and Canny, who derived a numerically robust technique to compute their roots in a few milliseconds.[30] Such methods can all be applied to solve any specialization of this loop and, in particular, those in Figure 3.

In recent years, resultant methods have also been used in Molecular Modeling. In their 1999 work,[2] for example, Wedemeyer and Scheraga use spherical geometry and elimination to derive resultants for the tripeptide, the seven-atom ring, and the disulfide bond loops. They obtain polynomials of degree 16, 16, and 32 for these three cases respectively. According to the previous paragraph, however, only the first two can be minimal-degree resultants. In fact, a 16th-degree resultant for the disulfide bond's equivalent robot (the 4-4 parallel manipulator) was already known by 1994,[25] and using the distance models of Figure 3, we readily see that 16 is the lowest possible degree in any case. This must be so, since Griffis and Duffy[26] proved that the six-atom ring can really adopt 16 possible conformations, and all other loops have this ring as a special case. Also, recent work by Coutsias et al.[4,5] finds that the standard Dixon and Sylvester resultants for the tripeptide loop have degree 16, which is in accordance with these observations.

In closing this section we highlight that, being reducible to a system of polynomial equations, the loop closure problem can in principle be solved by any general technique for such systems. Resultant-[31,32] and homotopy-based techniques[33] exist for that purpose, but they have a number of limitations in practice. On the one hand, resultant techniques only work for systems with zero-dimensional solution sets, usually introduce extraneous roots, and lead to eigenvalue problems of rapidly growing size. On the other, while advanced homotopy techniques can compute the irreducible decomposition of the solution set (which determines its connectivity), they are unable to isolate all of its points if its dimension is larger than one.

The method presented in this paper is a continuation of previous work in refs. 27, 34, and 35 and does not exhibit such limitations. In fact, the authors know of only one related technique that, as done in the paper, exploits the Bernstein form of the input equations within a similar branch-and-prune scheme.[36] Contrary to such technique, however, we directly formulate the equations in distance space, which allows the easy integration of NMR-derived bounds, and exploit the convex-hull property to its fullest extent, which yields a more accurate output, faster convergence, and the ability to isolate positive-dimensional solutions in lower times.

Computational Method

In Section Distance Models we rephrased the loop closure problem as one of finding all possible embeddings of a distance model. This section presents a method to compute all such embeddings. We first derive the loop equations that the embeddings must fulfill, then provide a complete root-finding algorithm for such equations, and finally analyze the overall computational cost.

Loop Closure Equations

Let $\mathbf{p}_1, \ldots, \mathbf{p}_k$, be k points in $\mathbb{R}^3$ and define the function

$$D(1,2,\ldots,k) = \begin{vmatrix} 0 & r_{1,2} & r_{1,3} & \ldots & r_{1,k} & 1 \\ r_{2,1} & 0 & r_{2,3} & \ldots & r_{2,k} & 1 \\ r_{3,1} & r_{3,2} & 0 & \ldots & r_{3,k} & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ r_{k,1} & r_{k,2} & r_{k,3} & \ldots & 0 & 1 \\ 1 & 1 & 1 & \ldots & 1 & 0 \end{vmatrix},$$

where $r_{i,j} = r_{j,i} = \|\mathbf{p}_i - \mathbf{p}_j\|^2$, i.e. the square distance between $\mathbf{p}_i$ and $\mathbf{p}_j$. $D(1,\ldots,k)$ is the general form of the Cayley–Menger determinant, initially used by Cayley in 1841,[37] but not systematically studied until 1928, when Menger showed its relevance in convexity analysis and other basic geometric problems.[38] Nowadays, this determinant plays a fundamental role in the Distance Geometry approach to molecular conformation.[6,7] To gain further insight, we recall its geometric interpretation for several cases.

If $k = 2$

$$D(1,2) = 2\, r_{1,2}.$$

If $k = 3$, and assumming $D(1, 2, 3) \leq 0$,

$$D(1,2,3) = -16A^2, \tag{1}$$

where A is the area of the triangle defined by $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. If $D(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ vanishes, the three points are collinear, and if it gives a positive value, the triangle cannot be assembled with the given distances.

If $k = 4$, and assuming $D(1, 2, 3, 4) \geq 0$,

$$D(1,2,3,4) = 288V^2, \tag{2}$$

where V is the volume of the tetrahedron defined by $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$. Similar to the $k = 3$ case, $D(1, 2, 3, 4)$ vanishes if and only if $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ lie on the same plane, and if it gives a negative value, then the tetrahedron cannot be constructed with the specified distances.

If $k > 4$,

$$D(1,\ldots,k) = 0 \tag{3}$$

because $D(1,\ldots,k)$ essentially gives the squared volume of the $(k-1)$-dimensional simplex defined by $\mathbf{p}_1,\ldots,\mathbf{p}_k$, but since this simplex is degenerate in $\mathbb{R}^3$, its volume must be zero.

If we have a set of points $\mathbf{p}_1,\ldots,\mathbf{p}_v$, and all distances between them are given, we can use conditions of this kind to check whether such a point configuration is embeddable in $\mathbb{R}^3$. The following theorem of Distance Geometry provides a set of necessary and sufficient conditions to this end.

The ν points $\mathbf{p}_1,\ldots,\mathbf{p}_v$ ($\nu > 3$) are embeddable in $\mathbb{R}^3$, satisfying the prescribed distances between them, if, and only if, there exist four of those points, say $\mathbf{p}_1,\ldots,\mathbf{p}_4$ without loss of generality, for which

$$D(1,2) > 0, \tag{4}$$

$$D(1,2,3) < 0, \tag{5}$$

$$D(1,2,3,4) > 0, \tag{6}$$

and for every pair ($\mathbf{p}_i$, $\mathbf{p}_j$), $i, j = 5, \ldots, \nu$, with $i < j$, we have

$$D(\mathbf{R}, i) = 0, \tag{7}$$

$$D(\mathbf{R}, j) = 0, \tag{8}$$

$$D(\mathbf{R}, i, j) = 0, \tag{9}$$

where $\mathbf{R}$ stands for the index sequence $1, \ldots, 4$.

While conditions (4)–(6) guarantee the embeddability of the tetrahedron defined by the points indexed in **R,** conditions (7)–(9) guarantee the compatibility of the distances between points in $\mathbf{R}$ and other point pairs ($\mathbf{p}_i$, $\mathbf{p}_j$) not in **R.** Observe that these conditions are indeed necessary, as $D(1,2)$ is twice the squared distance between $\mathbf{p}_1$ and $\mathbf{p}_2$, and, in accordance with eqs. (1)–(3), a Cayley–Menger determinant must be negative, positive, or strictly zero depending on whether it involves three, four, or more than four points, respectively. A proof that these are also sufficient conditions was given by Blumenthal in ref. 10.

The previous theorem provides a system of polynomial constraints whose solutions yield the possible spatial embeddings of a distance model: simply select any four points defining a non-coplanar tetrahedron, and collect inequalities (4)–(6) and eqs. (7)–(9). The fixed distances provide scalar entries in the Cayley–Menger determinants, and the unspecified ones are the system's unknowns, If, by some means, we are able to compute all possible values for such unknowns, then it is straightforward to derive a set of Cartesian coordinates for the points, using the method in ref. 39 for example.

Actually, Sippl and Scheraga showed that the previous system can be further simplified using Jacobi's theorem.[11] Assuming all conditions in eqs. (4)–(8) hold, eq. (9) can be substituted by

$$D^*(\mathbf{R}, i, j) = 0, \tag{10}$$

where

$$D^*(\mathbf{R}, i, j) = \begin{vmatrix} 0 & r_{1,2} & r_{1,3} & r_{1,4} & r_{1,i} & 1 \\ r_{2,1} & 0 & r_{2,3} & r_{2,4} & r_{2,i} & 1 \\ r_{3,1} & r_{3,2} & 0 & r_{3,4} & r_{3,i} & 1 \\ r_{4,1} & r_{4,2} & r_{4,3} & 0 & r_{4,i} & 1 \\ r_{1,j} & r_{2,j} & r_{3,j} & r_{4,j} & r_{i,j} & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{vmatrix}.$$

Having one less row and column, $D^*(\mathbf{R}, i, j)$ contains much less monomials (and usually less variables) than $D(\mathbf{R}, i, j)$, which yields faster evaluations in the algorithm below. Thus, we will use condition (10), in substitution of condition (9) above.

To summarize, the aforementioned conditions yield a polynomial system of the form

$$\mathbf{F}(\mathrm{x}) = 0, \quad \mathbf{G}(\mathrm{x}) > 0, \tag{11}$$

where $\mathbf{F} = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, $\mathbf{G} = (g_1(\mathbf{x}), \ldots, g_l(\mathbf{x}))$, $\mathbf{x} = (x_1, \ldots, x_n)$, and each function f_i, or g_i, is a Cayley–Menger or Sippl–Scheraga determinant involving the variables $x_1, \ldots, x_n$, which simply are new names for the unknown distances $r_{i,j}$ of the model. The new names and the re-indexing facilitate the explanations that follow, which describe a complete method to solve such equations efficiently.

A Complete Root-Finding Algorithm

This section provides an algorithm to isolate all solutions of System (11), based on the Bézier clipping technique developed in ref. 40. The algorithm receives System (11) as input, together with a box $\mathcal{B}$ of $\mathbb{R}^n$ where its solutions are to be sought for. This box is a rectangular domain of the form $\mathcal{B} = [a_1, b_1] \times \cdots \times [a_n, b_n]$, where $[a_i, b_i]$ denotes the closed real interval of possible values for the x_i variable. As output, the algorithm provides a collection of small subboxes of $\mathcal{B}$ bounding all solutions. The user can control the precision of the output by asking all box sides to be smaller than a threshold σ.

Observe that if δ is the sum of all known distances in the model, then all unknown distances can be bound to lie in the interval $[0, \delta^2]$, and we can set $\mathcal{B} = [0, \delta^2]^n$ initially. This trivial bound, however, can be further tightened using efficient bound-smoothing techniques such as the ones mentioned in refs. 41 and 42. Without loss of generality, however, we will assume that all variables in System (11) vary within the [0, 1] interval, so that $\mathcal{B}$ is actually the unit box $\mathcal{U} = [0, 1]^n$. Note that one can always perform an affine parameter transformation to the $\mathbf{x}$ variables,[†] to get $\mathcal{B}$ scaled to $\mathcal{U}$. This transformation also increases the numerical stability of the computations, as all variables now take values in the same interval.

Initially, the algorithm converts the polynomials in System (11) to the so-called Bernstein form. Using properties of such form, then, it reduces $\mathcal{U}$ as much as possible, by narrowing some of its defining intervals, to remove regions of $\mathcal{U}$ that contain no solution. The reduction is iterated until (1) the box is found to contain no solution, or (2) all box intervals are smaller than σ in width, or (3) the reduction is insignificant. In the latter case the box is split into two halves, and each of the halves is recursively reduced and split again in the same manner. The algorithm, in sum, follows a classic branch-and-prune scheme, but the pruning operation, as we will see, is quite elaborated.

The process is next described in detail. We first show how to translate System (11) to Bernstein form, then explain the root-finding procedure for a system with just one equation, next give its generalization to arbitrary polynomial systems, and finally show how it can be readily parallelized.

Conversion to Bernstein Form

Let m_i be the maximum degree of x_i in System (11) and define $M = (m_1, \ldots, m_n)$. It is well-known that $\mathbb{R}_M[\mathbf{x}]$, the set of polynomials in the variables $x_1, \ldots, x_n$ of degree $\leq m_i$ in x_i, forms a vector space. Usually, any polynomial $f(\mathbf{x})$ of System (11) will be expressed in the monomial basis of such space, in the form

$$f(\mathbf{x}) = \sum_{I=0}^{M} a_I x^I,$$

where the sum extends to all multiindex combinations from 0 to M (all $I = (i_1, \ldots, i_n)$ such that $0 \leq i_k \leq m_k$, for $k = 1, \ldots, n$) and x^I denotes the product $x_1^{i_1} x_2^{i_2}, \ldots, x_n^{i_n}$. The employed root-finding algorithm, however, requires these polynomials to be given in

[†]This can be done by applying a change of variables of the form $x_i' = (x_i - a_i)/(b_i - a_i)$ for $i = 1, \ldots, n$ on the input polynomials, using the multidimensional Horner scheme, for example.

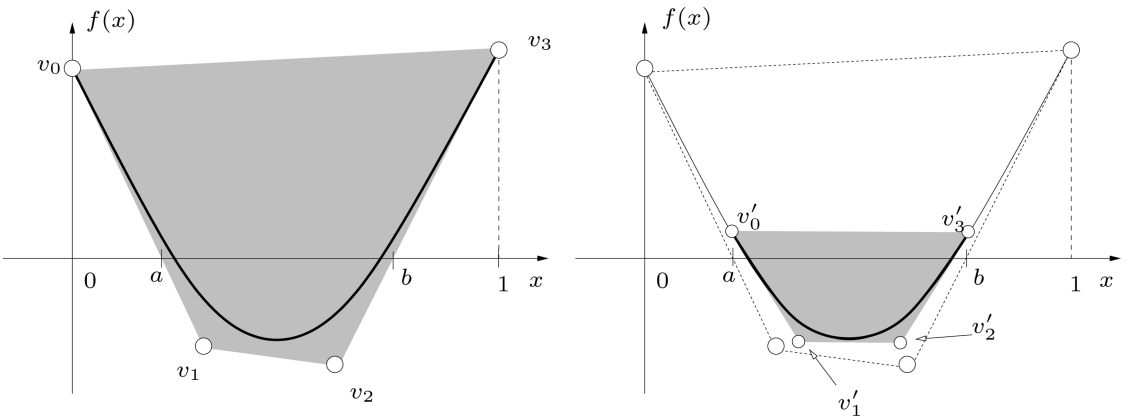

Figure 4. The convex-hull and subdivision properties illustrated for a planar curve of parametric form $(x, f(x)) = \Sigma_{i=0}^{3}\, v_i b_{i,3}$. Left: for $x \in [0, 1]$ the curve is inside the convex hull of the control points $v_0, \ldots, v_3$. Right: the control points v_i' corresponding to a subinterval $[a, b] \subset [0, 1]$ can be computed from the v_i's using the subdivision property. The v_i's are closer to $(x, f(x))$ than the v_i's.

the multivariate Bernstein basis,[43] formed by the multinomials $B_{I,M}(\mathbf{x})$, $I = 0, \ldots, M$, defined as

$$B_{I,M}(\mathbf{x}) = b_{i_1,m_1}(x_1) \cdots b_{i_n,m_n}(x_n),$$

where $b_{i,m}(x) = \binom{i}{m} x^i (1-x)^{m-i}$ denotes the ith Bernstein polynomial of degree m. Therefore, we have to convert every polynomial in (11) to the form

$$f(\mathbf{x}) = \sum_{I=0}^{M} c_I B_{I,M}(\mathbf{x}). \tag{12}$$

This expression is the Bernstein form of $f(\mathbf{x})$ and the coefficients c_I are called its *control points* relative to the unit box. The c_I can easily be computed from the a_I using the formula

$$c_I = \sum_{J=0}^{I} \frac{\binom{I}{J}}{\binom{M}{J}} a_J, \tag{13}$$

where the multiindex binomial coefficient for $I = (i_1, \ldots, i_n)$ and $J = (j_1, \ldots, j_n)$ is defined as

$$\binom{I}{J} = \binom{i_1}{j_1} \cdots \binom{i_n}{j_n}.$$

The Algorithm for One Equation

Let us assume for a moment that System (11) only contains one equation, say $f(\mathbf{x}) = 0$. This limitation will be removed later. To compute all of its solutions we first write $f(\mathbf{x})$ in Bernstein form, as in eq. (12). We then construct the function $F : \mathbb{R}^n \to \mathbb{R}^{n+1}$ defined as $F(\mathbf{x}) = (x_1, \ldots, x_n, f(\mathbf{x}))$. Clearly, finding the roots of $f(\mathbf{x})$ is equivalent to detecting all points of the form $(\mathbf{x},0)$ in the graph of $F(\mathbf{x})$. However, the latter formulation is advantageous. First, the graph of $F(\mathbf{x})$ is an algebraic variety in $\mathbb{R}^{n+1}$ whose points can be parameterized with polynomials in Bernstein form as

$$F(\mathbf{x}) = \sum_{I=0}^{M} v_I B_{I,M}(\mathbf{x}),$$

where $v_I = (i_1/m_1,\ i_2/m_2, \ldots, i_n/m_n,\ c_I)$ are the control points of $F(\mathbf{x})$,[40] relative to the unit box of $\mathbb{R}^{n+1}$. Second, to isolate all roots, we can use two useful properties of the Bernstein form of $F(\mathbf{x})$, illustrated in Figure 4:

- The *convex-hull* property dictates that when $\mathbf{x} \in [0,1]^n$, $F(\mathbf{x})$ is totally contained within the convex hull of its control points v_I. This follows immediately from the values taken by the Bernstein polynomials $B_{I,M}$ in the unit box. Since they are nonnegative and form a partition of unity,[43] the linear combination of control points v_I in $\Sigma_{I=0}^{M}\, v_I B_{I,M}(\mathbf{x})$ is actually a convex combination when $\mathbf{x} \in [0,1]^n$. Actually, it is equally possible to compute the control points of $F(\mathbf{x})$ relative to a sub-box $\mathcal{B} \subset [0,1]^n$ and, again, it will happen that their convex hull will enclose the graph of $F(\mathbf{x})$ for $\mathbf{x} \in \mathcal{B}'$.
- The *subdivision* property states that if we are interested in the values that $F(\mathbf{x})$ takes within a sub-box $\mathcal{B} \subset [0,1]^n$, then the control points relative to $\mathcal{B}$ can be computed from the control points relative to $[0,1]^n$ via the *Blossom* algorithm for Bézier patches.[43]

The important point to retain here is that, after subdivision, the new control points for $F(\mathbf{x})$ are closer to the graph of $F(\mathbf{x})$ than the previous ones (Fig. 4, right). These considerations permit the following recursive procedure to find all roots of $f(\mathbf{x})$ inside an arbitrary box $\mathcal{B} \subseteq [0,1]^n$:

1. *Initialization:* Compute the control points v_I of $F(\mathbf{x})$ relative to the unit box $[0, 1]^n$, using eq. (13).
2. *Box reduction:* Using the v_I's and the subdivision property, compute the control points of $F(\mathbf{x})$ relative to $\mathcal{B}$. Using the convex hull property, reduce $\mathcal{B}$ to a smaller sub-box $\mathcal{B}' \subseteq \mathcal{B}$ still containing all solution points of $f(\mathbf{x}) = 0$ within $\mathcal{B}$ (see the details below). If $\mathcal{B}$ gets reduced to an empty set then it contains no solution. In such case, mark $\mathcal{B}$ as an EMPTY box.
3. *Iterate reduction:* Set $\mathcal{B} = \mathcal{B}'$ and repeat Step 2 again until either (1) no significant reduction of $\mathcal{B}'$ with respect to $\mathcal{B}$ is achieved, or (2) $\mathcal{B}$ gets EMPTY.

4. *Convergence test and box splitting:* If $\mathcal{B}'$ is not EMPTY, see if it is "sufficiently" small. For this, just check whether its sides are shorter than a specified threshold σ. If it does, conclude that there is a root inside it, mark it as a SOLUTION box, and return $\mathcal{B}'$. Otherwise, split $\mathcal{B}'$ into two halves, yielding two equally-sized smaller boxes, and recursively apply Steps 2–4 on each of such boxes.

Note that, on termination, this process will have explored a binary tree of boxes whose internal nodes are boxes being split at some time, and whose leaves are either SOLUTION or EMPTY boxes. SOLUTION boxes form a discrete approximation of the solution set, as they enclose all of its points. Clearly, the smaller the σ, the more accurate this approximation will be.

It remains to be seen how the box reduction in Step 2 can be performed. Although there are several ways to implement it, leading to several variants of this algorithm, we restrict here to the most effective of them, that uses linear programming. Let $\mathcal{C} \subset \mathbb{R}^{n+1}$ denote the convex hull of the control points v_I, and let $\mathcal{R}$ be the region of intersection of $\mathcal{C}$ with the hyperplane $\{(\mathbf{x}, 0): \mathbf{x} \in \mathbb{R}^n\}$. Then, we define $\mathcal{B}'$ in Step 2 as the smallest rectangular box enclosing $\mathcal{R}$. Although the explicit computation of $\mathcal{R}$ is a complex and time-consuming task, it is not necessary to carry it out explicitly if all we need is just a bounding box for it. Indeed, $\mathcal{R}$ can be described with a set of linear equalities and inequalities as follows. Since any point $\mathbf{x} \in \mathcal{R}$ must be a convex combination of the control points v_I, there must be coefficients $\lambda_I \in \mathbb{R}$ such that

$$(\mathbf{x}, 0) = \sum_{I=0}^{M} \lambda_I \, v_I, \quad \lambda_I \geq 0 \ \ \forall I, \qquad \text{and} \ \ \sum_{I=0}^{M} \lambda_I = 1. \tag{14}$$

Then, to obtain the bounds of $\mathcal{B}'$ along dimension x_i we only need to maximize and minimize x_i, subject to the constraints in eqs. (14). These optimizations are linear programming problems and, hence, they can be efficiently solved via Simplex or interior-point methods. In sum, since there are n variables, the computation of $\mathcal{B}'$ in Step 2 involves solving a total of $2n$ linear programs.

The Algorithm for General Systems

The generalization of the previous strategy to the case of multiple equations and inequalities only requires slight modifications. First, we redefine $\mathcal{B}'$ in Step 2 as a sub-box of $\mathcal{B}$ that contains the simultaneous solutions of *all* equations and inequalities within $\mathcal{B}$. Second, to compute $\mathcal{B}'$, note that while the solutions of each individual equation $f(\mathbf{x}) = 0$ are bound to lie in a region defined by eqs. (14), the solutions of each individual inequality $g(\mathbf{x}) > 0$ are bound to lie in the convex region defined by

$$(\mathbf{x}, x_{n+1}) = \sum_{I=0}^{M} \mu_I \, u_I, \quad x_{n+1} \geq 0, \ \ \mu_I \geq 0 \ \ \forall I, \ \text{and} \ \ \sum_{I=0}^{M} \mu_I = 1, \tag{15}$$

where the μ_I are real coefficients, x_{n+1} refers to the last coordinate of $\mathbb{R}^{n+1}$, and the μ_I are the control points of $G(\mathbf{x}) = (\mathbf{x}, g(\mathbf{x}))$. Thus, to compute $\mathcal{B}'$, we simply need to minimize and maximize x_i, $i = 1, \ldots, n$, subject to all constraints of the form of (14) and (15) simultaneously, gathered for all equations and inequalities in System (11). As before, such linear programs can be solved using standard linear programming tools.

As it turns out, the previous strategy works for polynomial systems of any kind (either under-, well-, or over-constrained) with no modification. Actually, the algorithm usually performs better in overconstrained systems (i.e., with more equations than unknowns) than on equivalent well-constrained ones (with an equal number of equations and unknowns). This is because redundant equations introduce additional constraints in the linear programs to be solved, thus pruning larger portions of the search space at each iteration. This fact can be exploited by introducing more Cayley–Menger equations than those strictly induced by the reference tetrahedron.

Observe that the previous algorithm is *complete,* in the sense that there is a guarantee that every solution point will be contained in at least one SOLUTION box. Empirical tests show that it is also *correct,* meaning that, for a small enough σ, all SOLUTION boxes will contain at least one solution point. In other words, the output is free of the *cluster effect* observed in other branch-and-prune approaches.[27,44] Moreover, note also that, as the subdivision proceeds the roots of F can be moved away from their true positions because of the inevitable round-off errors. Nevertheless, Theorem 5 in ref. 45 ensures that the sensitivity of the roots versus small perturbations of the control points decreases monotonically under subdivision, which makes the algorithm quite robust in this sense.

Parallelization

The previous algorithm can be easily parallelized to be run on multiprocessor computers. To this end, we can just implement the book-keeping of the search tree on a selected "supervisor" processor, which at all times, keeps track of the tree leafs. Every leaf that is neither an EMPTY nor a SOLUTION box needs to be further reduced. Since box reduction is the most time-consuming task, and several boxes await for it simultaneously, it makes sense to perform the reductions in parallel, by assigning each of them to any of the remaining "child" processors. A child processor's task is thus to receive a box from the supervisor, to reduce it as much as possible solving the linear programs related to eqs. (14) and (15), and to return the reduced box back to the supervisor, which will queue it for further splitting and reduction, if needed, or mark it as SOLUTION or EMPTY otherwise.

Computational Cost

The computational cost of the algorithm can be analyzed by evaluating the cost of one iteration and the number of iterations it will have to perform, all in terms of the problem size. In our case, such size depends on two parameters essentially: the number of points (v) and known distances (e) in the model. Let us see how these parameters influence the two aforementioned factors.

Cost of One Iteration

We will assume that an iteration encompasses the application of Steps 2 to 4 to a particular box, and evaluate the worst-case cost of these steps using O-notation.

As for Step 2 (box reduction), note that it requires the solution of $2n$ linear programs, where n is the number of variables in the distance model. The best bound for the complexity of linear programming is due to Karmarkar,[46] who showed that the problem can be solved in $O(p^{3.5})$ time, where p is the number of variables intervening in the program. Thus, the total cost of Step 2 in terms of n and p is $C_{\text{Step 2}} = O(np^{3.5})$. To evaluate this cost in terms of v and e, we consider the worst possible case, which occurs when only the six distances in the reference tetrahedron **R** are known. In that situation, both n and p attain the largest possible value. Clearly, n reaches the peak value $n = \binom{v}{2} - 6 = O(v^2)$. To evaluate p, note that, in addition to the variables $x_1, \ldots, x_n$, the linear programs also involve the λ_I variables introduced by each Cayley–Menger or Sippl–Scheraga determinant in eqs. (14). For each determinant there are as many λ_I variables as control points its Bernstein form requires. Determinants of the type $D(\mathbf{R},i)$ are quadratic in each variable and involve four variables each, meaning they require 3^4 control points. Determinants of the type $D^*(\mathbf{R},i,j)$ are bilinear and involve nine variables each, meaning they require 2^9 control points. Since there are $v - 4$ equations of the former determinant type, and $\binom{v-4}{2}$ of the latter, we have $p = n + (v-4)3^4 + \binom{v-4}{2}2^9 = O(v^2)$, and thus we get $C_{\text{Step 2}} = O(v^9)$ in the worst case.

As for Steps 3 and 4, note that their operations (box copying, reduction checking, emptiness verification, convergence test, and splitting) only require visiting all box intervals once, and can hence be accomplished in time $C_{\text{Step 3}} = C_{\text{Step 4}} = O(n) = O(v^2)$. In conclusion, the overall cost of one iteration is $O(v^9)$ in the worst case, since it is dominated by the cost of the box reduction step.

Number of Iterations

It is difficult to predict how many steps the algorithm will require to isolate all solutions. The number of iterations largely depends on the chosen σ, and on the dimension d of the solution set. Clearly, the smaller the σ, the larger the number of box splittings and iterations required. Also, for a fixed σ, the amount of solution boxes grows exponentially with d. Thus, for a specific problem, an initial guess on the execution time is usually made on the basis of d only.

We can guess d from a global count of variables and constraints. Note that for v vertices we have a total of $3v - 6$ coordinates to be determined (the minus six corresponding to the elimination of global isometries by anchoring the reference tetrahedron to an absolute frame). Since there are e constraints, the dimension of the solution set should be $d = 3v - 6 - e$ in principle. This count, however, is a mere estimation, as it takes into account neither the connectivity graph of points and constraints in any way, nor the values of the known distances. Even if such graph had no overconstrained subgraph, the count could fail to predict d correctly.[12] It is worth mentioning, however, that if the distance model corresponds to a bond-bending network (the bond network of a molecule, plus extra distances locking the bond-bending angles), then there is strong evidence that the previous count is correct (assuming generic dimension values and no overconstrained parts). This is also conjectured for a corresponding count in terms of the number of rigid bodies (b) and free torsion angles (h) of such model, $d = 6b - 6 - 5h$.[12] A formal proof of these statements is still lacking, however.

Extra information on the algorithm's speed can be drawn from analyzing the local convergence properties. It has been proven that a branch-and-prune scheme such as the one presented is quadratically convergent to each root if $d = 0$.[40] Intuitively, this means that once the algorithm is able to produce a list of boxes containing a solution each, the error committed when approximating a solution by any point of its enclosing box decreases quadratically on subsequent iterations. Thus, when the solution space is zero-dimensional, the algorithm exhibits asymptotic performance similar to that of fast single-root finding procedures such as Newton-Raphson. Being a multiroot finder, however, the algorithm will converge to all roots, and it is well known that, yet in, the zero-dimensional case, the number of such roots is exponential with v,[47] which renders the number of iterations at least exponential with v. Finally, we mention that the determination of the convergence rate for $d > 0$ is still an open problem, but our empirical tests indicate that the algorithm converges linearly to the roots if $d = 1$, and sublinearly if $d \geq 2$.

Study Cases

The algorithm has been implemented in C. The current program employs the parallel processing scheme explained previously, implemented via the Message Passing Interface library,[48] and the optimization problems involved in box reduction are solved with the Simplex method implementation provided by the GLPK package.[49]

The program has been successfully tested on single- and multiprocessor machines. We next illustrate its performance on rigid loops, mobile loops, and multiloop molecules. In all cases, the algorithm outputs a collection of small boxes enclosing all points of the conformational space. In rigid loops, the boxes are isolated, with one box for each possible conformation. In mobile ones, they form groups of adjacent boxes approximating the different connected components of the conformational space. Such components correspond to continuous self-motions of the molecule that maintain loop closure (also known as pseudo-rotation paths), which can be easily simulated. Except on one indicated case (where super-computation is used), all CPU times will be given for a single-processor Intel Xeon PC, running at 3 GHz under Linux. Detailed numerical output of all experiments can be obtained by contacting the authors.

Closing Rigid Loops

Rigid loops have zero-dimensional conformational spaces. They can only adopt a finite number of conformations, which corresponds to a number of isolated points in distance space. To test the algorithm on such cases, we have applied it to the disulfide bond, the tripeptide, and the seven-atom loops. The chosen instances of such problems respectively had 18, 4, and 3 possible solutions. The algorithm correctly found them in 825, 193,

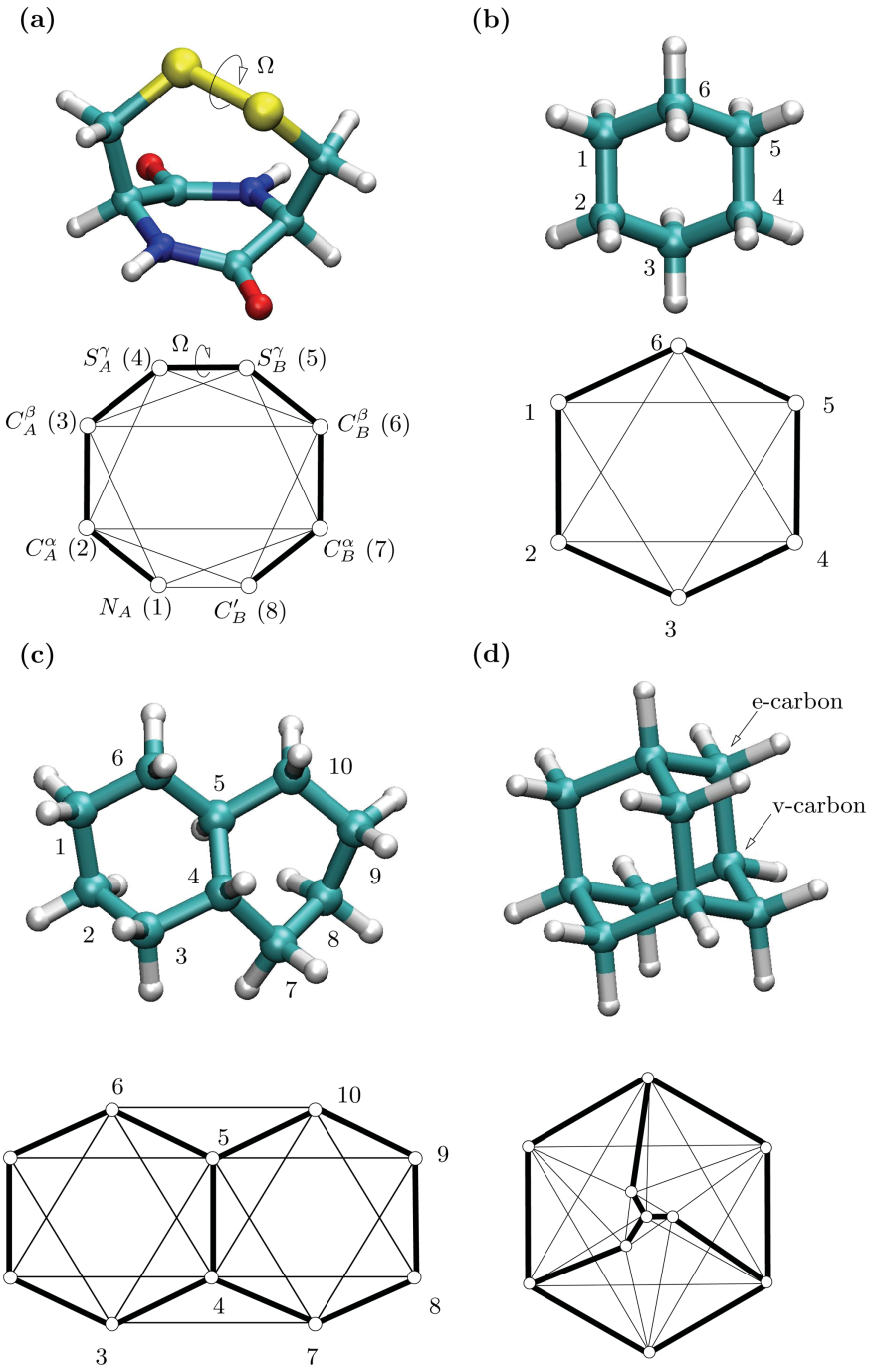

Figure 5. Some of the analyzed loops and their distance models. (a) The disulfide bond. (b) Cyclohexane. (c) Bicyclohexane. (d) Adamantane. In all models, a line joining two atoms indicates that the distance between them is fixed. Thick lines correspond to covalent bonds. Except Ω in (a), all torsion angles are *a priori* unknown. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

and 2.8 s, respectively, with the σ threshold set to 0.001 Å^2 in all cases. Although it has not been checked, these times probably compare unfavorably with those of specific methods such as the ones discussed in refs. 1 and 2, which being based on resultants, should be faster. To have an idea, previous work in Robotics[30] reports execution times of a few milliseconds when solving the resultant of the most general $s = 6$ loop, which has the mentioned three loops as special cases. However, we note that the strength of the presented method is not its speed but, rather, its ability to deal with arbitrary single- or multiloop structures.

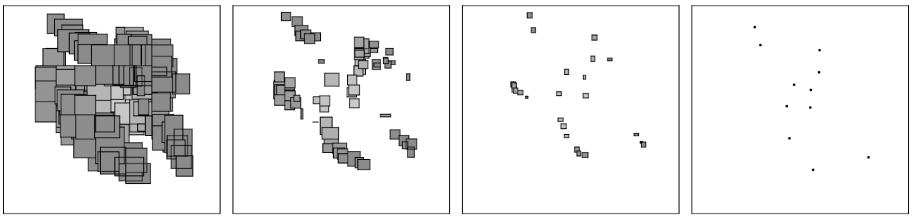

Figure 6. Four stages of the root search performed on the disulfide-bond loop. The x and y axes correspond to $r_{2,5}$ and $r_{3,7}$, respectively.

We shall work out the disulfide bond in detail. The tripeptide and seven-atom loops are formulated and solved in an analogous way. Figure 5a depicts this loop and the distance model we employ. The model fixes the same atom–atom distances as the one in ref. 2. It is assumed that the 3D coordinates of N_A, C_A^α, C_B^α, and C_B' are held fixed, while the positions of the four other atoms remain to be determined. All torsion angles are to be determined too, except Ω, which is fixed to 90° and determines the distance between the two C^β atoms. The geometric parameters we chose yield

$$\begin{aligned}
r_{1,2} &= 2.1050, & r_{7,8} &= 2.3892, & r_{5,7} &= 7.9244,\\
r_{2,3} &= 2.4106, & r_{8,1} &= 57.3503, & r_{6,8} &= 6.6307,\\
r_{3,4} &= 3.2178, & r_{1,3} &= 6.1279, & r_{3,6} &= 16.6496,\\
r_{4,5} &= 4.1544, & r_{2,4} &= 7.9290, & r_{1,7} &= 39.8786,\\
r_{5,6} &= 3.2133, & r_{3,5} &= 10.0384, & r_{2,8} &= 45.0087,\\
r_{6,7} &= 2.4105, & r_{4,6} &= 10.0348, & r_{2,7} &= 31.7352,
\end{aligned}$$

where $r_{i,j}$ denotes the squared distance between atoms i and j, in square angstrom. The remaining distances, $r_{4,8}$, $r_{1,6}$, $r_{4,7}$, $r_{1,5}$, $r_{1,4}$, $r_{5,8}$, $r_{2,5}$, $r_{2,6}$, $r_{3,7}$, and $r_{3,8}$ are to be determined by the algorithm.

To collect the embeddability conditions for this loop, we choose the reference tetrahedron **R** defined by atoms N_A, C_A^α, C_B^α, and C_B'. In this case, since the distances within **R** are all known and consistent, eqs. (4)–(6) trivially hold and only eqs. (7), (8), and (10) are relevant. Then, the minimal system to be solved is

$$\begin{aligned}
&D(1,2,7,8,3) = 0, && D^*(1,2,7,8,3,4) = 0,\\
&D(1,2,7,8,4) = 0, && D^*(1,2,7,8,3,5) = 0,\\
&D(1,2,7,8,5) = 0, && D^*(1,2,7,8,3,6) = 0,\\
&D(1,2,7,8,6) = 0, && D^*(1,2,7,8,4,5) = 0,\\
& && D^*(1,2,7,8,4,6) = 0,\\
& && D^*(1,2,7,8,5,6) = 0.
\end{aligned}$$

Figure 6 illustrates how the algorithm iteratively bounds all roots of these equations. Since there are 10 distances to be determined, the search space is 10-dimensional. The initial box $\mathcal{B}$ is defined by setting all its intervals to [0, 40] Å^2. The figure depicts the boxes created during intermediate stages of the algorithm, projected on the $r_{2,5}$–$r_{3,7}$ plane. As shown, by iterating box reduction and box splitting, the roots get progressively bounded with more and more precision (Figs. 6a–6c), until only

Table 1. Solutions of the Disulfide-Bond Loop.

$r_{4,8}$	$r_{1,6}$	$r_{4,7}$	$r_{1,5}$	$r_{1,4}$	$r_{5,8}$	$r_{2,5}$	$r_{2,6}$	$r_{3,7}$	$r_{3,8}$
29.696	25.081	18.420	23.724	11.519	13.114	15.321	17.362	30.259	42.602
31.559	29.822	20.465	15.252	9.262	17.530	13.227	23.474	26.735	37.174
28.402	25.027	19.428	13.653	11.504	16.928	10.623	17.352	30.923	41.689
18.105	36.516	12.201	30.660	16.092	11.755	21.018	28.985	21.509	32.861
18.769	32.039	13.541	28.523	16.005	10.706	20.212	26.005	24.031	34.299
25.771	38.441	14.755	27.928	11.186	13.418	19.967	31.104	16.673	27.036
27.899	29.104	21.053	21.223	16.486	11.945	12.019	21.114	29.363	41.620
26.216	26.432	20.905	20.111	16.255	12.143	11.304	19.438	30.635	41.376
25.010	26.870	13.454	13.764	10.202	17.654	11.521	19.979	23.882	35.542
28.012	25.149	20.966	23.178	16.296	13.675	15.346	17.394	30.476	42.840
26.249	25.117	20.960	23.150	16.292	11.821	15.333	17.377	30.495	41.224
27.267	25.199	18.347	23.785	11.512	11.361	15.369	17.423	30.199	40.903
24.739	38.449	14.756	23.222	12.412	17.430	20.024	31.113	16.674	27.039
24.669	36.620	14.756	21.139	10.336	17.415	19.907	31.092	16.674	26.952
22.756	37.995	11.317	23.831	9.546	17.590	20.920	30.611	19.478	30.525
21.889	36.646	10.770	21.299	9.689	17.442	20.111	31.121	17.894	27.998
25.708	36.646	14.761	25.311	9.311	13.367	20.103	31.121	16.681	26.953
22.203	32.682	11.238	17.495	9.534	17.476	14.038	24.850	19.273	29.363

All values are in square angstroms.

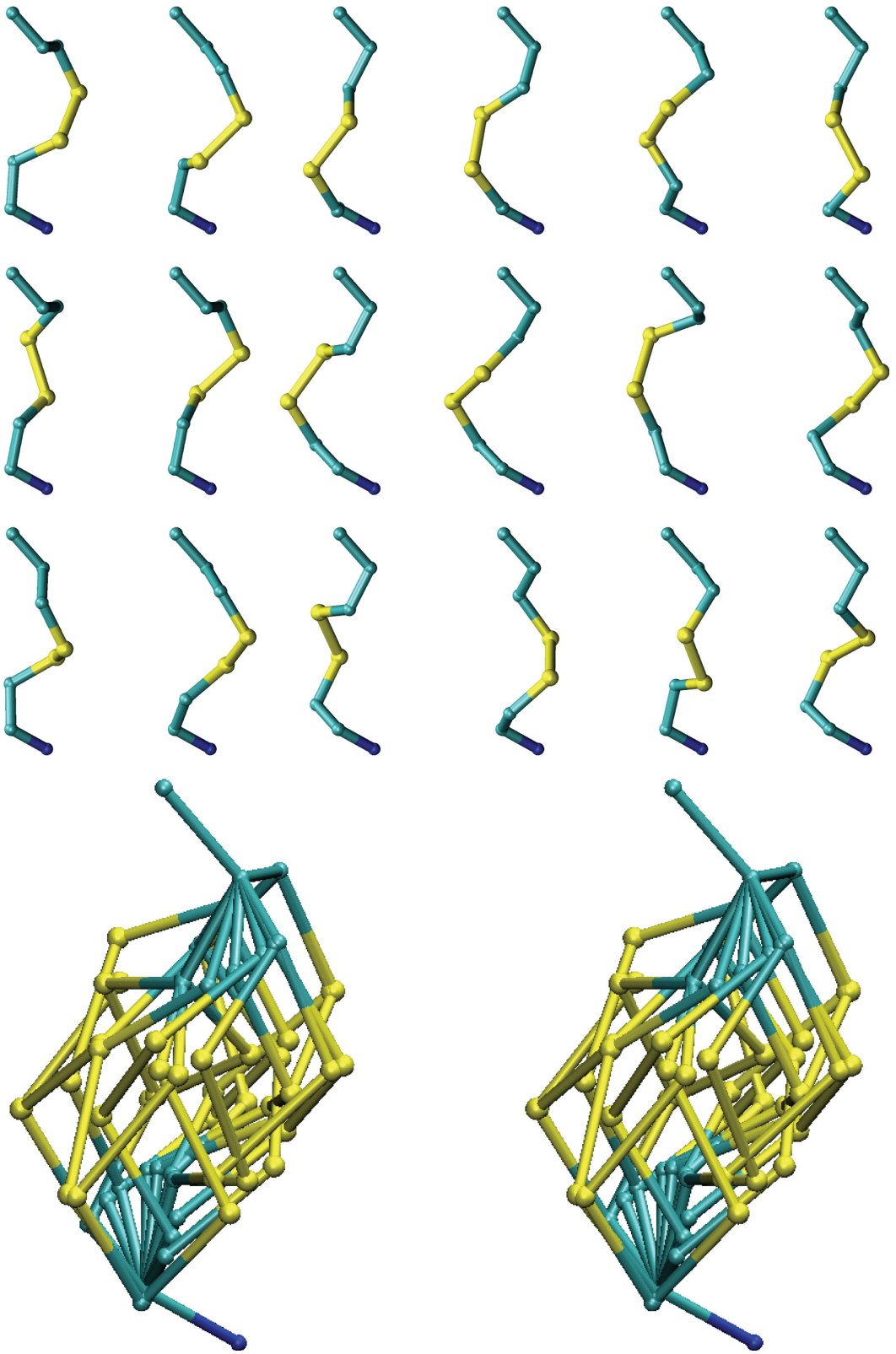

Figure 7. The 18 conformations of the chosen disulfide bond loop, for $\Omega = \pm 90^{\circ}$. The bottom row shows a stereogram of all conformations overlaid. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

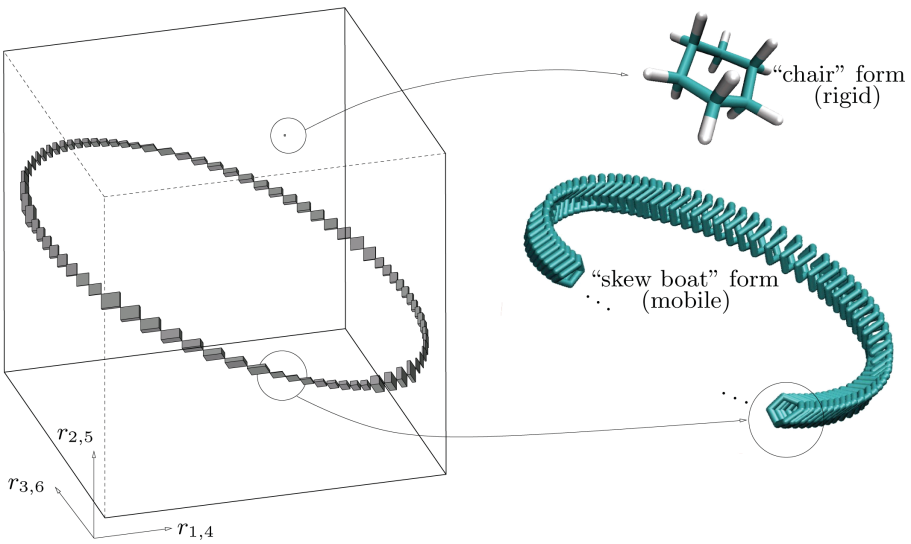

Figure 8. The conformational space of cyclohexane has one isolated point and a cyclic one-dimensional path, corresponding to the chair and skew boat conformations, respectively. The intervals of the shown bounding box are [6.1, 9.3] Å^2 in all dimensions. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

one box per solution remains in the end (Fig. 6d). A total of 18 solutions are found, listed in Table1, which, as shown, project onto 11 points on the $r_{2,5}$–$r_{3,7}$ plane.

The solutions must be properly interpreted. Since a distance model cannot distinguish between a given conformation and its specular image (all pairwise distances will be the same in both cases), each of the output boxes actually corresponds to two possible Euclidean embeddings of such model. Thus, the 18 boxes correspond to 36 different conformations, divided into two groups of 18 conformations each, having positive and negative orientation on the reference tetrahedron. The positively oriented ones are depicted in Figure 7. Nine of them correspond to solutions for $\Omega = 90^\circ$, and the other nine for $\Omega = -90^\circ$.

Closing Mobile Loops

The algorithm was also tested on cyclohexane, cycloheptane, and cyclooctane, which have conformational spaces of dimension 1, 1, and 2, respectively. The spaces of the first two molecules were already mapped out completely by Crippen using linearized embedding,[50] and our results are in agreement with his. It is worth mentioning that, while our technique is complete (it guarantees that all solutions can be found) Crippen's method relies on an initial grid sampling, and hence may fail to find all solutions if the grid is not fine enough. Regarding cyclooctane, to our knowledge, this is the first time that a complete accurate map of its conformational space is obtained. As admitted by Crippen, cyclooctane's surface is complicated to describe analytically and computationally difficult to map out.[50] Linearized embedding could be used to isolate its conformational space, but the obtained map would not be complete in the sense defined in this paper. Moreover, other previous work on this molecule just focuses on finding low-energy conformers with sampling and local search techniques.[51,52] The map we provide is exhaustive (it contains all points of the conformational space) and may thus be used to determine its topology, or derive the whole (guaranteed) network of minimum-energy conformers and saddle point transitions between them.

Figure 5b illustrates how such single-loop cycloalkanes can be encoded as distance models. We only provide cyclohexane's model, but models of larger carbon rings are analogous. Since all bond lengths and bond angles are fixed, and all torsion angles can vary, the model is simply a closed sequence of "carbon triangles," pairwise sharing an edge. Carbon atoms are numbered 1, 2, 3, ... clockwise, following their order in the ring. The number of variable distances to be determined is 3 in cyclohexane, 7 in cycloheptane, and 12 in cyclooctane. Note that hydrogen atoms need not be encoded in the models because they impose no closure condition, and their position can be determined once the conformation for the carbon ring is obtained.

For these loops, the reference **R** may be defined on three consecutive carbons and one of the hydrogen atoms bonded to the middle one of such carbons. This, however, obliges introducing this hydrogen as part of the distance model, which increases the number of unknowns in the problem. An alternative is to identify four carbons in the ring whose enclosed volume never vanishes, and gather the equations such reference generates. However, a four-tuple with this property does not exist in these rings. (The boat form of cyclohexane is a conformation where this volume gets zero, for example.) On cyclohexane, nevertheless, the two 4-tuples defined by four consecutive atoms, $i, \ldots, i+3$, and the next four, $i+1, \ldots, i+4$, never get coplanar simultaneously. On cycloheptane and cyclooctane, this happens for three consecutive such tuples.[‡] Thus, we may gather the equations all of these refer-

[‡]This can be verified with the implemented software by solving System (11) for these molecules, with the Cayley–Menger determinants of such tetrahedra equated to zero.

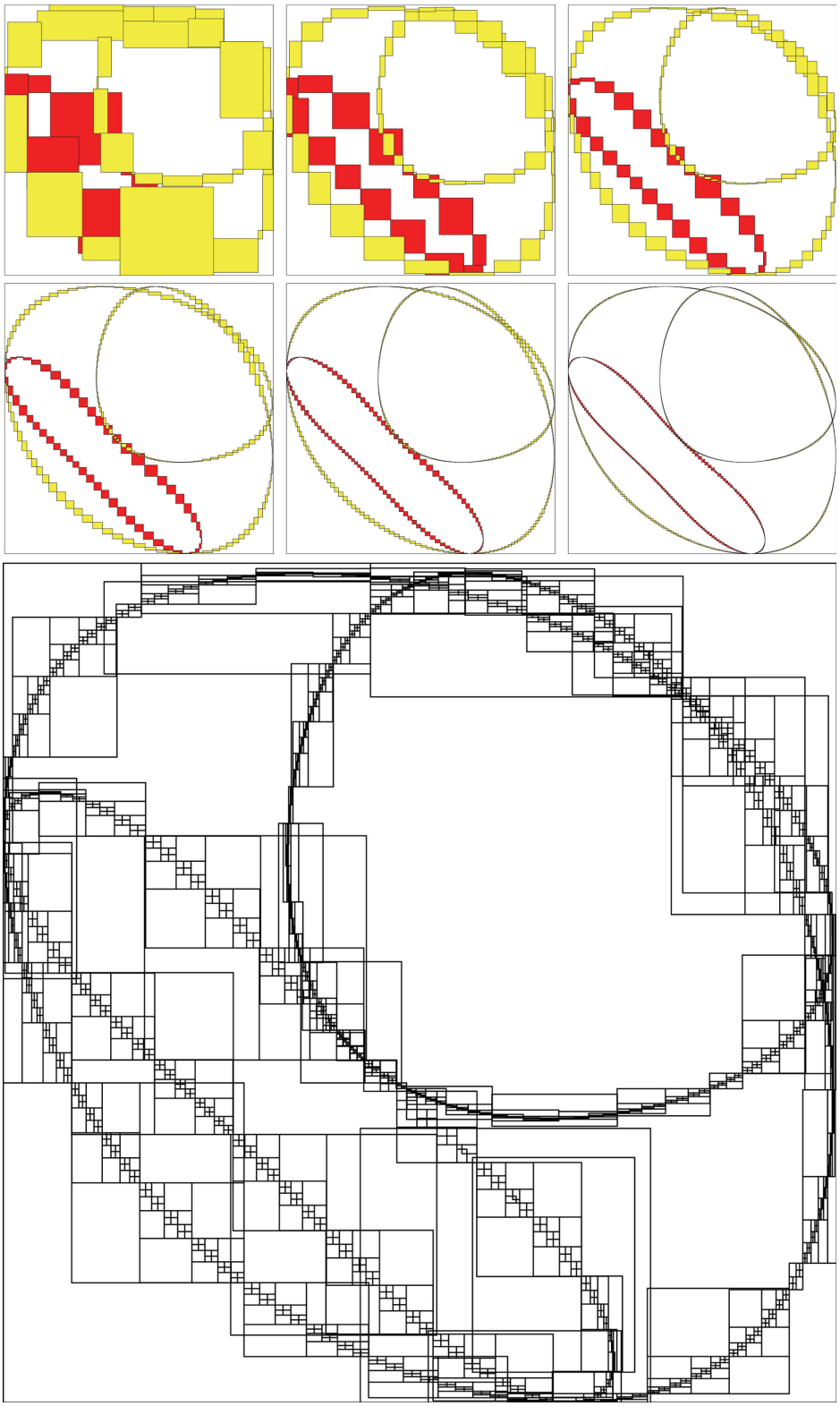

Figure 9. Top and middle: The conformational space of cycloheptane obtained at different precisions. Red and yellow boxes correspond to the chair and boat pseudo-rotation paths. Bottom: Box splittings and reductions performed by the algorithm to obtain the rightmost plot of middle row. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

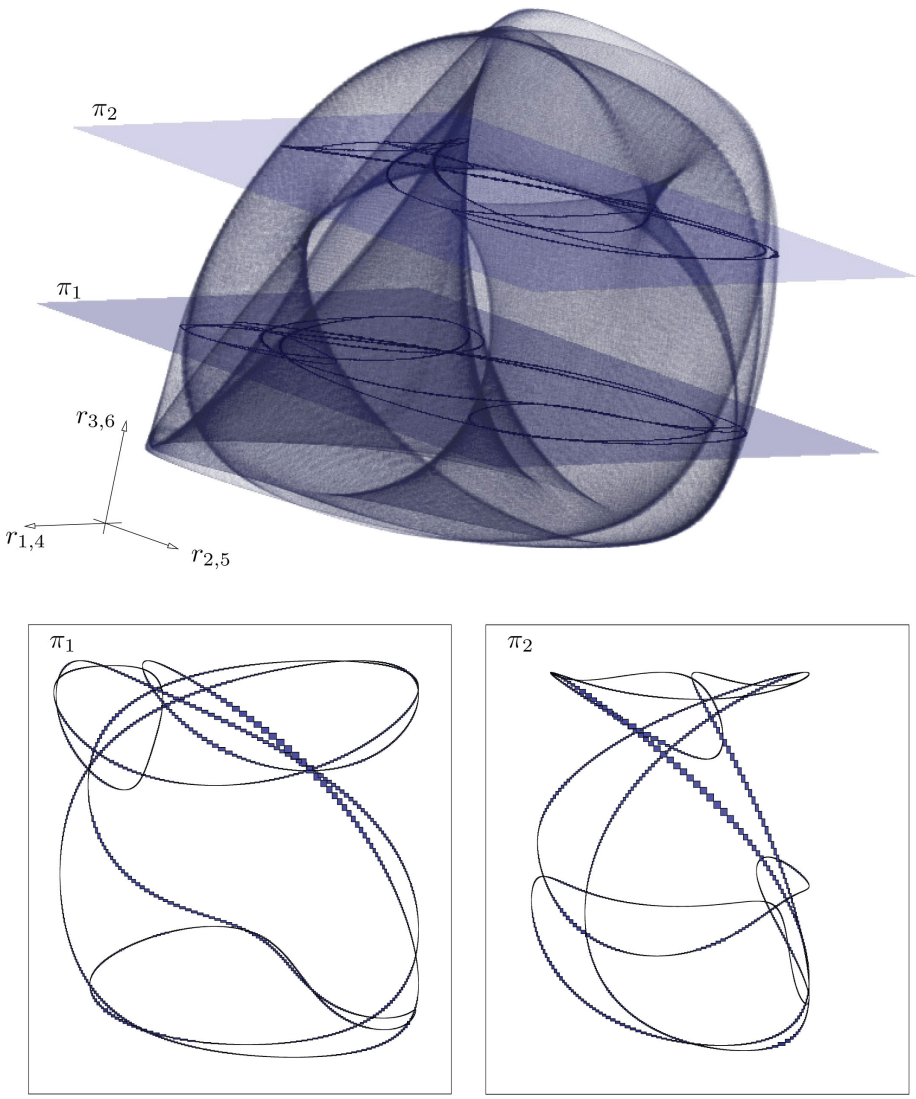

Figure 10. Top: Boxes approximating the conformational space of cyclooctane, plotted for their $r_{1,4}$, $r_{2,5}$, and $r_{3,6}$ dimensions. The approximation contains 273,626 boxes, which are here shown with semitransparent walls. Bottom: Two slices of such space, corresponding to clipping the surface with the planes π_1 and π_2 shown above. The left slice corresponds to fixing $r_{3,6} = 8.5$ Å^2, and the right one to $r_{3,6} = 11.5$ Å^2. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

ences generate, saving, for a ring with c carbons, $c - 3$ variables with respect to the previous formulation.

Figure 8 shows the output boxes returned by the algorithm on the cyclohexane loop, plotted in the space defined by $r_{1,4}$, $r_{2,5}$, and $r_{3,6}$. The problem was solved in 1.16 s, at $\sigma = 0.1$ Å^2. As expected, the conformational space is formed by an isolated point plus a closed curve of mobile conformations, corresponding to the "chair" and "skew boat" forms of this molecule, respectively. For clarity, hydrogen atoms are omitted in the boat-form conformations of the figure.

The output boxes for cycloheptane are shown in Figure 9, computed at decreasing σ values, and plotted on the $r_{1,4} - r_{3,7}$ plane. The figure also illustrates the sequence of box splittings and reductions performed by the algorithm, during the computation of the rightmost plot in the middle row. Using three consecutive four-tuples of carbons as references, this problem involves 18 equations in seven unknowns and was solved in 11.2 min, at $\sigma = 0.1$ Å^2. From the output, we see that the conformational space is formed by two cyclic curves (shown in yellow and red). They correspond to the two possible forms of this molecule (the chair and the boat) which are mobile and exhibit a one-dimensional pseudo-rotation path. Although the plots seem to indicate so, the curves do not intersect, but this can only be realized by checking the adjacency relationships of the actual seven-dimensional boxes. This corresponds to the known fact that one cannot convert the chair to the boat form without deforming the bond lengths and angles.

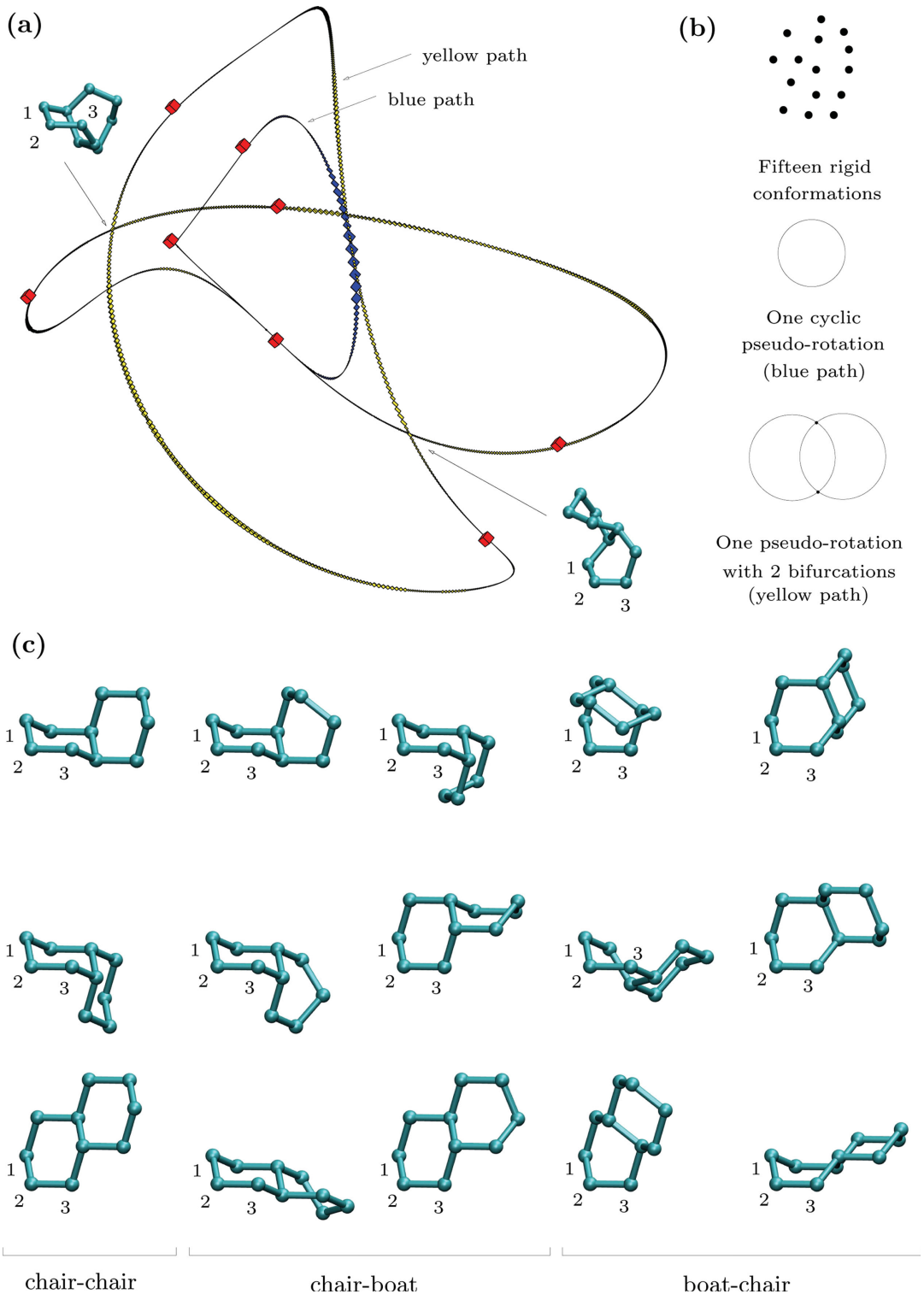

Figure 11. (a) Boxes approximating the conformational space of bicyclohexane. Red boxes correspond to rigid conformations. They are somewhat enlarged to appreciate them. Yellow and blue boxes correspond to the two mobile boat–boat forms. Two bifurcation points exist on the yellow path, corresponding to the shown twisted boat–boat conformations. (b) The actual topology of the space. (c) The fifteen rigid conformations, with atoms 1, 2, and 3 held in a fixed position. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Table 2. Summary of the experiments.

Study case	v	e	n	b_e	b_s	n_e	s_t	n_{br}	t_{br}	t_{tot}
Disulfide	8	18	10	525	18	20	120 × 570	4045	0.20	825
Tripeptide	9	21	15	13	4	45	279 × 1410	121	1.59	193
7 atom	7	15	6	67	3	6	30 × 81	423	0.0062	2.8
Cyclohexane	6	12	3	231	116	6	23 × 63	553	0.0021	1.16
Cycloheptane	7	14	7	1,671	651	15	79 × 445	4073	0.16	672
Cyclooctane	8	16	12	554,717	273,626	58	379 × 7122	1,710,778	4.2	7,185,268
Bicyclohexane	10	25	20	6,441	2,219	41	238 × 848	25,968	0.92	23,978
Adamantane	10	12	15	25	1	63	414 × 5146	202	5.86	1,184

All times are given in CPU seconds, for an Intel Xeon at 3 GHz. The time t_{tot} for cyclooctane is estimated, as the actual experiment was carried out in the MareNostrum computer grid.[53] See the text for details.

We had to rely on supercomputation to obtain a complete map for cyclooctane. Certainly, although the minimal formulation only involves 20 equations in 12 unknowns, being a two-dimensional hypersurface, the number of boxes approximating this space is huge for $\sigma = 0.1$ Å^2. The estimated time to compute this approximation is 83 days on a Pentium Xeon PC at 3 GHz, but we were able to complete it in 72 hours using the parallelized algorithm on 128 processors of the Barcelona MareNostrum computer grid.[53] A total of 273,626 solution boxes were obtained, constituting (to the authors' knowledge) the most comprehensive description to date of this molecule's conformational space. Figure 10 offers a view of these boxes, projected onto the $r_{1,4}$, $r_{2,5}$, and $r_{3,6}$ dimensions. To realize the complex topology of the approximated space, we have drawn the boxes with semitransparent walls. A highly contorted surface arises, with many self-intersections and singularity points, as can be appreciated from the two isodistance curves of such surface given in the figure. By analyzing all boxes, we found that they form a single connected component, meaning this molecule can move from any two conformations without distorting all bond lengths and angles. This contradicts previous conjectures stating that cyclooctane's space could have two or more connected components.[50,54]

Closing Multiple Intermingled Loops

The presented methodology can also be used on multiloop structures. We refer here to interconnected loops where the motion of any loop influences the motion of the others, so that the analysis of the overall conformational space is irreducible to the analysis of the individual loop spaces. To illustrate the method's performance on them, we will employ it to derive the feasible conformations of bicyclohexane and adamantane. The former molecule serves testing the performance on a complex conformational space (with isolated conformations, one-dimensional paths, and bifurcation points), and both problems demonstrate the method's utility on finding the feasible conformations of a multiloop molecule without resorting to Dreiding models or energy-minimization techniques.

The bicyclohexane molecule can be seen as two cyclohexane loops sharing one torsion angle. Its distance model is given in Figure 5c. Here, we defined the reference tetrahedron on carbons 3, 4, 5, and 7. Overall, the minimal system involves 21 equations in 20 unknowns. Figure 11a shows its solution boxes, plotted on the space of $r_{3,8}$, $r_{3,10}$, and $r_{4,7}$. They were computed for $\sigma = 0.1$ Å^2 in 6.6 hours on the above-mentioned desktop PC. A detailed analysis of the output reveals that the boxes form 17 connected components, 15 of which are isolated boxes, and the remaining two are one-dimensional box paths (shown in yellow and blue in the figure). While the isolated boxes yield the indicated 15 rigid conformations (3 chair-chair, 6 boat-chair, and 6 chair-boat forms), the one-dimensional paths correspond to two different pseudo-rotations of the molecule (with the two rings in boat form). Interestingly, the yellow path actually exhibits two bifurcation points with four branches each. (Other apparent crossings are not true bifurcations when analyzed in the full 20-dimensional distance space.) In other words, when the molecule attains one of such points, its motion can continue in either of four ways, as the figure indicates. In sum, the topology of the conformational space is as depicted in Figure 11b: 15 isolated points, two one-dimensional paths, one of which containing two bifurcations connected by four branches.

The adamantane molecule can be seen as four cyclohexane loops, arranged as shown in Figure 5d. Topologically, the network of covalent bonds forms a tetrahedron, with four carbons placed on its vertices and six carbons placed on its edges. For convenience, we call these two carbon types *v-carbons* and *e-carbons,* respectively. To formulate the loop closure equations, we can employ eqs. (7), (8), and (10), with reference to the tetrahedron defined by any v-carbon and its three neighboring atoms. This yields a minimal system of 21 equations in 15 unknowns, but if we solve it considering all bond angles set to the standard sp^3 hybridization value, we rapidly find that it has no solution. This is because, being all overlapped, the loops impose more distance constraints than really needed to fix the spatial positions of all atoms. Technically, the molecule is said to be "redundantly rigid" or "overconstrained," which means that almost all bond angles render the loop-closure constraints unsatisfied. Angles satisfying the loop constraints can be readily computed by the algorithm, however. By symmetry, we may assume the angles (though unknown) are equal on all v- and e-carbons, and we may replace their corresponding distances in the equations by a single indeterminate variable. In this situation, the 21 equations contain one more variable and yield one

solution (in 19.7 min), confirming that the only possible conformation of adamantane under these constraints is the one in Figure 5d.

As a summary, Table2 provides several statistics illustrating the algorithm's computational cost. For each case, v is the number of points in the model, e the number of known distances, n the number of variables, b_e the total amount of explored boxes, b_s the number of solution boxes, n_e the number of equations and inequalities (the minimal set, plus redundant equations to speed up the convergence), s_t the size of the Simplex tableau (rows $\times$ columns), n_{br} the number of "box reduction" steps performed, t_{br} the average time needed to complete one box reduction, and t_{tot} the total time required to solve the problem. All times are given in seconds.

Conclusions

We have presented a method to derive complete maps of molecular loop conformational spaces. Such maps are given as box approximations enclosing all points of the conformational space, and can be obtained at any desired precision. This means the method is *exact,* in the sense used in refs. 2 and 3, since the returned conformations satisfy all loop closure constraints up to an error that can be made arbitrarily small. Moreover, the paper has shown that the method is also *general* (it can deal with single or multiple loops interconnected in an arbitrary way) and *complete* (it converges to all possible solutions, even if they form positive-dimensional sets).

The method encodes a multiloop structure as a distance model, and computes the unknown distances of such model using a branch-and-prune scheme. At its core, this scheme employs a new procedure to obtain feasibility intervals for each unknown distance, assuming that the remaining distances vary within known intervals. Since we enforce the compatibility of all distances with the Cayley–Menger equations involving up to six points, this procedure can be seen as a generalization of previous bound-smoothing processes such as the ones mentioned in refs. 41 and 42, which only enforce the compatibility with triangle or tetrangle inequalities (i.e., the Cayley–Menger equations involving three or four points, respectively).

The method's performance has been illustrated on conformational spaces with dimensions ranging from zero to two. Its convergence rate is quadratic for zero-dimensional spaces, linear for one-dimensional spaces, and sublinear otherwise. Thus, the algorithm is practical for low dimensions, but it clearly suffers from the *curse of dimensionality,* since the amount of boxes needed to approximate a space grows exponentially with its dimension. However, the fact that the algorithm is easily parallelizable allows alleviating this problem. We have shown, for example, that the challenging two-dimensional space of cyclooctane, which could only be approximated by sampling techniques before, can now be accurately isolated using a grid computer of moderate size. Experiments with the algorithm have proved, for the first time, that this space is simply connected, an information that could not be confirmed with sampling methods. The algorithm could still be useful on larger spaces, if combined with some sampling technique. A simple way to do so would be to let the algorithm compute a low-resolution approximation of the conformational space (as done in the first two plots of Fig. 9) and then launch a stochastic or grid sampling method confined to the returned boxes. Very likely, this would increase the probability of convergence of the local searches performed by the sampling. This point clearly deserves further attention.

It is worth mentioning that there are classes of distance models which can be efficiently embedded in a constructive fashion, without resorting to iterative techniques.[7,8] In general, thus, it would be convenient to decompose any model into constructable and nonconstructable parts, in order to apply the presented solver only to the latter. Although this point requires further research, the fact that nonconstructable distance models of arbitrary size exist implies that a general solver will always require an iterative technique of some sort. The paper's emphasis, thus, has been on providing one such technique.

We remark finally that the loop closure problem arises in several domains, on any situation where all postures of a ring of hinged bodies need to be known. In Molecular Modeling such bodies are groups of atoms with fixed relative positions, but the paper has shown that we also encounter them in Robotics, as the rigid links of serial and parallel manipulators. Important techniques for loop closure have been given from within both fields, but techniques obtained by one community frequently remain unnoticed by the other, probably due to the lack of previous work clarifying the correspondences between molecular and robotic loops. To help filling this gap, we have made an effort not only to survey the main advances in both fields, but also to summarize the principal robot-molecule correspondences, and to provide tools to detect others.

Acknowledgments

Part of the experiments were carried out at the MareNostrum computer of the Barcelona Supercomputer Center, within the project "Isolating Configuration Spaces of Polycyclic Robots and Molecules." The authors are grateful to this center for granting access to their machines. The authors wish to thank the reviewers of the paper for their fruitful comments.

References

1. Gō, N.; Scheraga, H. A. Macromolecules 1970, 3(2), 178.
2. Wedemeyer, W. J.; Scheraga, H. J Comput Chem 1999, 20(8), 819.
3. Gibson, K. D.; Scheraga, H. A. J Comput Chem 1997, 18(3), 403.
4. Coutsias, E. A.; Seok, C.; Jacobson, M. P.; Dill, K. A. J Comput Chem 2004, 25(4), 510.
5. Coutsias, E. A.; Seok, C.; Jacobson, M. P.; Dill, K. A. Int J Quant Chem 2006, 106(1), 176.
6. Crippen, G. M.; Havel, T. F. Distance Geometry and Molecular Conformation; Research Studies Press: Taunton, England, 1988.
7. Havel, T. F. In Encyclopedia of Computational Chemistry, von Ragué, P.; Schreiner, P. R.; Allinger, N. L.; Clark, T.; Gasteiger, J.; Kollman, P. A.; Schaefer, H. F., III. Eds.; Wiley, 1998; pp. 723–742.
8. Porta, J. M.; Ros, L.; Thomas, F. In Proceedings of the IEEE International Conference on Robotics and Automation; 2005; pp. 960–967.
9. Saxe, J. B. In Proceedings of the 17th Allerton Conference in Communications, Control and Computing, 1979; pp. 480–489.

10. Blumenthal, L. M. Theory and Applications of Distance Geometry; Oxford University Press, 1953.
11. Sippl, M. J.; Scheraga, H. Proc Nat Acad Sci 1986, 83, 2283.
12. Whiteley, W. Phys Biol 2005, 2, 116.
13. Flexweb server. http://flexweb.asu.edu.
14. Jacobs, D.; Rader, A. J.; Kuhn, L.; Thorpe, M. Proteins 2001, 44, 150.
15. Thorpe, M.; Lei, M.; Rader, A. J.; Jacobs, D. J.; Kuhn, L. J Mol Graph Model 2001, 19, 60.
16. ProFlex web page. http://www.bch.msu.edu/labs/kuhn/web/software.html.
17. Gō, N.; Scheraga, H. A. J Chem Phys 1969, 51, 4751.
18. Pieper, D. The Kinematics of Manipulators under Computer Control; Ph.D. Thesis; Stanford University, 1968.
19. Roth, B.; Rastegar, J.; Scheinman, V. In On the Theory and Practice of Robot Manipulators; First CISM IFTOMM Symposium; 1973; pp. 93–113.
20. Shenkin, P. S.; Yarmush, D. L.; Fine, R. M.; Wang, H. J.; Levinthal, C. Biopolymers 1987, 26, 2053.
21. Hong, M.; Scheraga, H. A. Encyclopedia of Computational Chemistry, Vol. 5; Wiley, 1998; pp. 552–556.
22. Canutescu, A. A.; Dunbrack, R. L. Prot Sci 2003, 12, 963.
23. Lee, H.-Y.; Liang, C.-G. Mech Mach Theor 1988, 23(3), 209.
24. Raghavan, M.; Roth, B. Trans ASME J Mech Des 1993, 115, 502.
25. Lin, W.; Duffy, J.; Griffis, M. ASME J Mech Des 1994, 114, 444.
26. Griffis, M.; Duffy, J. J Robotic Syst 1989, 6, 703.
27. Porta, J. M.; Ros, L.; Thomas, F.; Torras, C. IEEE Trans Robot 2005, 21(2), 176.
28. Tsai, L.-W.; Morgan, A. ASME J Mech Transm Autom Des 1985, 107, 189.
29. Primrose, E. J. F. Mech Mach Theor 1986, 21(6), 509.
30. Manocha, D.; Canny, J. IEEE Trans Robot Autom 1994, 10(5), 648.
31. Dixon, A. L. Proc Lond Math Soc 1908, 6(2), 468.
32. Emiris, I. Z.; Canny, J. J Symbolic Comput 1995, 20(2), 117.
33. Sommese, A. J.; Wampler, C. W. The Numerical Solution of Systems of Polynomials Arising in Engineering and Science; World Scientific, 2005.
34. Bombín, C.; Ros, L.; Thomas, F. In Advances in Robot Kinematics; Lenarci, J.; Stanisic, M. M.; Eds; Kluwer, 2000; pp. 53–60.
35. Bombín, C.; Ros, L.; Thomas, F. In Proceedings of the IEEE International Conference on Robotics and Automation, 2001; Vol. 3; pp. 3332–3337.
36. Zhang, M.; White, R. A.; Wang, L.; Goldman, R.; Kavraki, L.; Hassett, B. Bioinformatics 2005, 21(5), 624.
37. Cayley, A. Camb Math J 1841, II, 267.
38. Menger, K. Am J Math 1931, 53, 721.
39. Sippl, M. J.; Scheraga, H. Proc Nat Acad Sci 1985, 82, 2197.
40. Sherbrooke, E. C.; Patrikalakis, N. M. Comput Aided Geomet Des 1993, 10, 379.
41. Dress, A. W. M.; Havel, T. Discrete Appl Math 1988, 19, 129.
42. Rajan K.; Deo N. Bull Math Biol 1999, 61(5), 987.
43. Farin, G. Curves and Surfaces for Computer Aided Geometric Design—A Practical Guide; Academic Press, 1990.
44. Morgan, A.; Shapiro, V. ACM Trans Math Software 1987, 13(2), 152.
45. Farouki, R. T.; Rajan, V. T. Comput Aided Geomet Des 1987, 10, 379.
46. Karmarkar, N. K. Combinatorica 1984, 4, 373.
47. Borcea, C.; Streinu, I. Discrete Comput Geometry 2004, 31(2), 287.
48. Message Passing Interface web page. http://www-unix.mcs.anl.gov/mpi/index.htm.
49. Makhorin, A.; GLPK—the GNU linear programming toolkit. http://www.gnu.org/software/glpk.
50. Crippen, G. J Comput Chem 1992, 13(3), 351.
51. Kolossváry, I.; Guida, W. J Am Chem Soc 1993, 115, 2107.
52. Rocha, W. R.; Pliego, J. R.; Resende, S. M.; Dos Santos, H. F.; De Oliveira, M. A.; De Almeida, W. B. J Comput Chem 1998, 19(5), 524.
53. Barcelona Supercomputing Center web page. http://www.bsc.es.
54. Emiris, I. Z.; Mourrain, B. Algorithmica 1999, 25(2–3), 372.