A low resolution model for protein structure prediction from NMR data

Olivier Perriquet, Marco Correia, Pedro Barahona, Ludwig Krippahl

Centro de Inteligência Artificial - Departamento de Informática Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa 2829-516 Caparica PORTUGAL
{operriquet,mvc,pb,ludi}@di.fct.unl.pt

Abstract. We address in this paper the problem of protein structure prediction from Nuclear Magnetic Resonance data (NMR). We revisit the constraint programming approach we initiated in [KB02] and define a simpler model, where only the protein backbone is considered, which has the obvious advantage of dealing with one order of magnitude fewer atoms but also poses other problems, namely a less constrained problem, where the structure of the backbone defined by the existing constraints is less defined when compared with the full protein model. In this paper we describe our findings, concentrating on the alternative modelings that we have been addressing, together with the preliminary results obtained.

1 Introduction

Protein structure prediction is a fundamental problem in Bioinformatics, since it is well known that the 3D shape of a protein strongly determines the ligands (other proteins, viruses, smaller molecules as drugs, etc) with which it may interact, and hence strongly determines its functioning.

Notwithstanding the research on ab initio methods that aim at finding protein shape from first principles (minimization of some energy function), with simplified models (e.g. the lattice model [HL92]) or by connecting homologue components (Rosetta [KRB99]), other methods aim at integrating experimental data (either from X-Ray crystallography, or Nuclear Magnetic Resonance-NMR) in the determination of the protein structure.

This was the option we have been adopting with PSICO [KB02] by developing a constraint programming approach to handle the distance constraints provided by NMR experiments. Although competitive with other methods that use meta-heuristics local search alone [GMW97], the propagation of all the distance constraints is quite fast but not as precise. In fact, improving the initial solutions through constraint propagation alone is very costly, especially when all the atoms of the protein are considered.

Therefore, we have been testing a simpler model, where only the protein backbone (in fact the alpha-carbons of the protein residues) are considered. This has the obvious advantage of dealing with fewer atoms (one order of magnitude fewer) which is of great importance in combinatorial problems. Nonetheless, it

poses other problems, namely a less constrained problem, where the structure of the backbone defined by the existing constraints is less defined when compared with the full protein model.

In this paper we describe our findings, concentrating on the alternative modelings that we have been addressing, together with the preliminary results obtained. We first recall the main features of the model used so far (some formalization is provided to describe the latest improvements). Then we present our simpler model and its promising behavior on experimental data sets.

2 Improvements on the PSICO algorithm

We start with a brief overview of the PSICO (Processing Structural Information with Constraint propagation and Optimization) algorithm. More details can be found in [KB02]. The first stage of the PSICO algorithm builds a molecular structure that is an approximate solution to a set of geometric constraints. This structure is generated by reducing the three-dimensional domains representing the possible positions of the atoms until all domains are smaller than a threshold value in all dimensions (typically 2Å), or until there is excessive backtracking (typically a upper limit of 100 backtracking steps). This approximate solution is then refined in the second stage by a local search optimization algorithm. The idea is to provide the local search algorithm with the best possible initial guess without increasing computation time.

The geometrical constraints can be distance constraints between atom pairs [KB02] or constraints on the angles or relative placements of atoms in rigid groups [KB03], but this paper considers only the case of pairwise distance constraints. In this framework the constraints are propagated by either reducing the size of an atom domain or by inserting exclusion volumes inside the domain region. To make this process efficient and tractable the domains are represented by sets of cuboid volumes. One cuboid represents the boundaries of the domain (*Good* region), and a set of zero or more non-overlapping cuboids contained within the larger cuboid represents the exclusion volumes (*NoGoods* set) which the atom is known not to occupy. Constraints on the upper limit of the distance between two atoms are propagated by reducing the *Good* region. Constraints on the lower limit of the distance between two atoms are propagated by adding cuboids to the *NoGoods* set.

This representation of the domains has the useful property of preserving the shape of its elements, since all intersections of cuboids result in either empty domains or cuboids. However, this requires the distance constraints to be converted from the standard Euclidean formulation, where a distance to a point defines a sphere in three dimensions. Instead of this sphere, a constraint on the upper limits of the distance is considered to be the smallest cube containing the Euclidean distance sphere, whereas a constraint on the lower distance limits is considered to be the largest cube contained in the sphere. This weakens the constraints but ensures that no correct solutions are excluded by propagation.

Propagation is achieved by enforcing arc-consistency on these cuboid domains with this constraint representation. Once the system is arc-consistent, there is a partial enumeration step where one domain is split in two equal parts across its longest dimension. One part is discarded and arc-consistency is again enforced to propagate this change. The domains are selected for splitting in a round-robin schedule, whereby each domain is selected once before any domain is selected again. During each cycle the smallest domains are selected first (a first-fail approach), but all domains that are below the size threshold are excluded from this enumeration procedure. Backtracking involves selecting the alternative half of each domain.

The local search could start from any configuration but its performance clearly depends on the accuracy of the guess used as a starting point. When the first stage is cut short due to excessive backtracking, the local search is provided with a configuration that may be quite distant from the true structure and therefore as meaningful as a random start. In the following we first provide a mathematical formalization which may be considered a guideline of all the improvements we did, then we show the origin of this excessive backtracking and how our tentatives to overcome the problem led us to change the modeling itself.

2.1 Approximating spheres (l_2) by spheres (l_∞)

Mathematical context – Given two subsets A and A' of an affine space, we define the sum $A + A'$ as $\{a + a', a \in A, a' \in A'\}$ (Fig. 1). Later everything happens in the affine space $\mathbb{R}^3$. The notation $\overline{A}$ refers to the complementary set of A in $\mathbb{R}^3$. We call $Sphere(\delta)$ the three dimensional sphere of radius δ and origin 0 (the sphere for the Euclidean norm l_2), and $Cube(\delta)$ the cube of edge length 2δ centered in 0 (the sphere for the norm l_∞). We shall be using the usual approximation of l_2 by l_∞ and two straightforward properties of the sum:

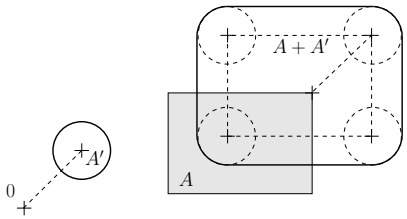


Fig. 1. The sum $A + A' = \{a + a', a \in A, a' \in A'\}$ of two affine subsets.

[**norm**]: $Cube(\delta/\sqrt{3}) \subset Sphere(\delta) \subset Cube(\delta)$
[**incl**]: $if\ B \subset C\ then\ (A + B) \subset (A + C)$
[**cubo**]: *Cuboids are stable both under intersection and sum*

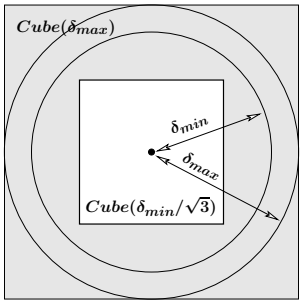
From the definition, it follows that:

$$dist_{l_2}(a,b) \geq \delta \iff a \in (\{b\} + \overline{Sphere(\delta)})$$
$$dist_{l_2}(a,b) \leq \delta \iff a \in (\{b\} + Sphere(\delta))$$

And:

$$dist_{l_2}(a,b) \geq \delta \; for \; all \; b \in B \iff a \in (B + \overline{Sphere(\delta)})$$
$$dist_{l_2}(a,b) \leq \delta \; for \; all \; b \in B \iff a \in (B + Sphere(\delta))$$

Modeling – A Euclidean constraint between 2 atoms is a pair $\Delta = (\delta_{min}, \delta_{max})$ which denotes the minimum and maximum distances allowed between them. The domain of an atom a_i is defined as $Dom_i = Good_i \cap NoGoods_i$ where $Good_i$ is a cuboid and $\overline{NoGoods_i}$ is a union of cuboids to be removed. The initial $Good_i$ are the whole space (in fact an arbitrary large cuboid), the initial $NoGoods_i$ are empty. The *In* constraints refer to δ_{max}, the *Out* constraints to δ_{min}.



$$Cube(\delta_{min}/\sqrt{3}) \subset Sphere(\delta_{min}) \subset Sphere(\delta_{max}) \subset Cube(\delta_{max})$$

Fig. 2. Approximation of two embedded Euclidean spheres by cuboids.

Selecting the values – The selection is made by bisection on the *Good* regions. We use a First Fail heuristic (prune as soon as possible the search tree) by bisecting across the largest dimension among x, y, z, and selecting the half which is the less likely to contain the atom (the half that intersect the most the other domains).

Propagating the* In *constraints – If we were strictly following the Euclidean model, the directional propagation of the *In* constraint from a $Good_i$ region to another $Good_j$ region should be: $Good_j \longleftarrow Good_j \cap (Good_i + Sphere(\delta_{max}))$. In our simplified model, however, $Good_j$ was set as a cuboid at the beginning and we want it to stay a cuboid. Thus we use the properties [**incl**], [**cubo**] and [**norm**] (see the approximation of two embedded spheres Fig. 2) to overestimate a bit the reduction of $Good_j$: $Good_j \longleftarrow Good_j \cap (Good_i + Cube(\delta_{max}))$.

Propagating the* Out *constraints – Following the same idea (the *NoGoods* regions must stay the complement of a list of cuboids), we underestimate the propagation of *Out* constraints: $NoGoods_j \longleftarrow NoGoods_j \cap (Good_i + \overline{Sphere(\delta_{min})})$

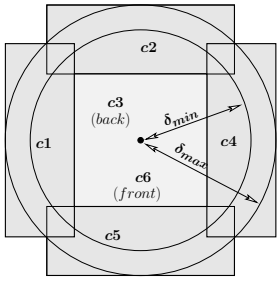
by the slightly smaller: $NoGoods_j \longleftarrow NoGoods_j \cap (Good_i + \overline{Cube(\delta_{min}/\sqrt{3})})$. With **[norm]**, **[incl]**, **[cubo]**, it follows that $\overline{NoGoods_j}$ stays a union of cuboids. These transformation rules in hand, we handle them with a classical arc propagation algorithm [Mac75]. During the propagation of *NoGoods* , the algorithm needs to ensure they do not overlap to be able to compute the total volume, check if the domain is empty, and fail when it is. Both the list of *NoGoods* and their scope during propagation should be bounded to avoid a combinatorial explosion.

2.2 Discussion about over-backtracking

We usually observe a propensity for backtracking and we believe this behavior has two causes. In this section we identify the first one and describe how we worked the problem. In the next section, we propose an alternative modeling to address the second cause. The propagation rules for *In* and *Out* constraints ensure the local consistency *within the cuboid model*, which is a relaxation of the actual distance geometry problem. Once a solution is found, we have the guarantee that it is globally consistent within the cuboid model, although it may not fulfill the initial Euclidean restraints. When running the solver on a small polyhedron (tens of atoms), the model quickly provides a solution that is consistent within the simplified model, but not in the Euclidean one. The cuboid approximation increases the slack on the actual Euclidean constraints, with the corners of the cuboids allowing solutions that would be forbidden by the constraints. The simplest way to correct this is to test the Euclidean constraints during propagation:

if $Good_j \cap (Good_i + Sphere(\delta_{max}))$ is empty **then** fail
else $Good_j \longleftarrow Good_j \cap (Good_i + Cube(\delta_{max}))$ **end if**

However, this test increases backtracking, especially at the later stages of the enumeration. This and the added expense of propagating the *Out* constraints itself makes the computation impractical. To overcome the problem, we designed an alternative representation of forbidden regions, where we drop the dual view *Good* vs. *NoGoods* to consider the domains as (possibly) *non convex*.



$CuboidList(\Delta, 6) = \cup_{k=1..6}\ c_k$

Fig. 3. Approximation of the embedded Euclidean spheres by a cuboid list (a collection of possibly overlapping cuboids). Here is represented the approximation for a list of size $N = 6$, which is obviously a minimum to model the hole inside the spheres.

Modeling – The use of *NoGoods* regions is equivalent to a modeling of non convex domains (the *NoGoods* regions are actually digging holes into the *Good* region). Instead of removing forbidden parts from the cuboid which represents the current domain, we model the domain by a union of (possibly overlapping) cuboids $Dom_i = \cup_{k=1}^{N} Cuboid_{i,k}$. A common bound N is fixed on the number of cuboids (typically $N = 6$ is a minimum to approximate an Euclidean sphere, as may be seen on Fig. 3). This new representation offers a better approximation of the Euclidean constraints and reduces the previous artifact (hence a fail may occur sooner). Furthermore, we neither need to ensure the non overlapping of the *NoGoods* regions nor to separate the propagation of *In* and *Out* constraints.

Selecting the values – We select either one cuboid if the size of the list is greater than one, or we bisect the remaining cuboid if the list is a singleton. Our default heuristic is a First Fail choice, as previously.

Constraints propagation – The definition of sum is extended in a natural way to a collection of sets: $\{A_i\}_{i\in I} + \{A'_j\}_{j\in J} := \{A_i + A'_j\}_{(i,j)\in I\times J}$ (see Fig. 4).

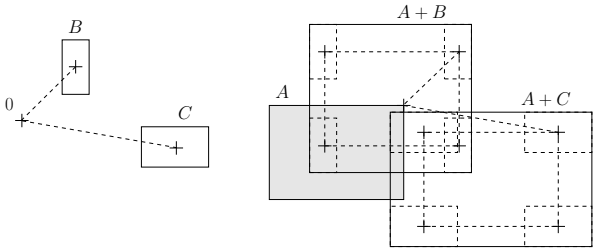


Fig. 4. The sum $A + \{B, C\} = \{A + B, A + C\}$. Note that we are really considering a collection of sets, not a union (B and C may overlap for instance).

During the propagation step, the cuboids of two lists are pairwise compared and we regularly take the convex hull of some cuboids formed during the process in order to keep the list size bounded by N.

Algorithm 1 - Propagate (Domain D_j, Domain D_i, Constraint Δ)

```
for all Cuboid c in D_j do
  TempList ⟵ ∅
  for all Cuboid c' in D_i do
    TempList ⟵ [ TempList, c ∩ (c' + CuboidList(Δ, N)) ]
  end for
  c ⟵ ConvexHull (TempList)
end for
```

Although the handling of lists obviously slows down the process, the propagation of *Out* constraints is made more efficiently and therefore can be deployed all along search. The improvement should also be seen through a more theoretical

scope. The model actually brings a new feature: the union of cuboids represents a disjunction of constraints, and is adequate to model *constraint ambiguity*. It also offers a larger choice than a simple bisection for the value selection. This feature can help design better heuristics and improve the search of a solution. In the next section we show how we improve the model in this manner and we discuss the uniqueness of the solution.

3 Improving the model: back to discrete domains

Since the exact position of the atoms are not assigned during search, and we have, for each atom, only a domain to be reduced, the heuristics we designed are based on the properties of these domains, namely the volumes of intersecting cuboids. A wrong guess early in the search often leads to unacceptable backtracking. To overcome the problem we focus on the modeling itself to help design more precise heuristics, especially at the beginning of the search process, when the atoms are dispatched in space. We assume that a nearly correct positioning of the protein backbone is enough to start the second stage and in the following we work at the backbone level, where only the C_α atoms are modeled. This has the obvious advantage of dealing with much less atoms.

3.1 The model

Discrete modeling – If n is the length of the sequence of residues of the protein, we define the variables $[x_i] = (x_1, x_2, x_3, .., x_n)$ as the 3D positions of the corresponding C_α in the sequence. The protein is assumed to remain within a cube of a given predefined size, this cube is discretized in p^3 small cells, providing the same finite domain for all the variables: $D_i = [1..p]^3$. Each value of these domains is called a cell, the size of a cell is chosen in order to avoid two C_α being in the same cell (2Å is convenient, with regard to the minimum distance between two C_α in any protein), and p is arbitrarily fixed according to the assumption of the diameter of the molecule. The molecule may be seen as discretized to a low resolution and the underlying distance geometry problem *projected* to this low resolution. Once a solution is found (every domain being reduced to a singleton), we assign to the corresponding variable the position of the center of the remaining cell. Note however that our model is different from the lattice model [Wil02], as we assume during search that the atom can be anywhere in the cell.

Selecting the values – Considering a structure solution, there are different ways to embed it into the low resolution model. Even if three C_α are fixed (to break all the Euclidean symmetries), this triangle of atoms would still have some looseness, due to the size of the cells, and the constraints themselves are somehow looser than the actual Euclidean constraints (see Fig. 5). Hence, a single solution of the actual problem, admits many representations in the model. This looseness endows the model with the interesting property that the solutions can be reached

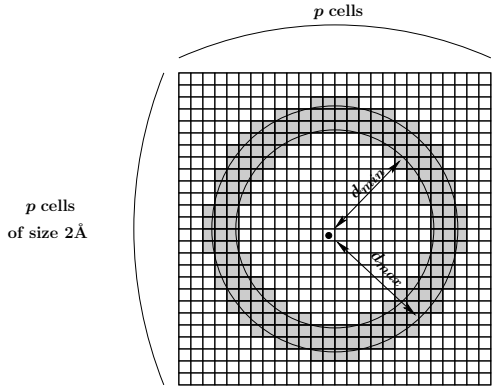


Fig. 5. Approximation of the embedded spheres in the discrete model (we drew the median section in 2D). The domain of a C_α, built up with p^3 cells, reduces to the gray cells under a constraint $\Delta = (\delta_{min}, \delta_{max})$.

straightforwardly or with little backtracking. This property justifies our choice for the selection heuristics: we select the variables by decreasing order of domain size and label the values randomly.

Propagating – Arc consistency is computed with an adaptation of AC6 [Bes94]. Note that on some test samples (see the experimental results in the next section), forward checking was enough to find the solution with almost no backtracking. Note that the number of variables ($\sim 10^2$) is here very small compared to the size of the domains ($\sim 10^4$).

3.2 Performances of the model on different networks

For all the tests we used the same protocol. The proteins of the test set are actual proteins from which the structure is known (the exact atomic positions stored in a PDB file). The features that are really meaningful with regard to our solver are their length and diameter.

The protocol – We calculate the distances between all pairs of C_α from the known positions of the atoms (the PDB files), and simulate different constraint networks, by either adding slack on the full graph, or decreasing thickness, or both. Our aim is to test the robustness of the model with regard to these relaxations. The final target is a graph that really simulates NMR data (only some short range distances with a slack). The default cell size is 2Å. Two C_α are fixed at the beginning: the C_α we assume closest to the center is fixed in the center cell of its domain, and its next neighbor on the backbone is fixed two cells apart. The two first propagation steps are only forward checking. In all the forthcoming examples, the average is taken on 20 independent runs, and for each run, a timeout is fixed at 10 min. All the distances are in Å, all the time measurements in seconds.

sample (reference)	diam.	n	p	$\frac{n \times p^3}{1000}$	RMSD min : **avg** : max ± **stdev**	TIME **avg**
smpl-1 (RCSB011828)	25.1 Å	41	15	138	1.77 : **1.98** : 2.15 ± **0.11**	**0.25**
smpl-5 (1KOE.part)	31.7 Å	50	18	292	1.77 : **2.07** : 2.29 ± **0.13**	**0.95**
smpl-2 (RCSB009932)	35.2 Å	71	20	568	1.60 : **1.74** : 1.93 ± **0.09**	**2.30**
smpl-8 (1BNL.part)	39.5 Å	60	22	639	1.94 : **2.20** : 2.47 ± **0.17**	**1.70**
smpl-4 (RCSB001509)	37.2 Å	87	21	806	1.58 : **1.72** : 1.82 ± **0.06**	**7.20**
smpl-7 (RCSB016762)	38.3 Å	97	22	1033	1.57 : **1.65** : 1.84 ± **0.07**	**5.40**
smpl-3 (RCSB009965)	38.3 Å	124	22	1320	1.51 : **1.57** : 1.69 ± **0.05**	**9.80**
smpl-6 (RCSB015705)	44.8 Å	92	25	1438	1.70 : **1.87** : 2.06 ± **0.12**	**4.95**

Table 1. Experimental results with the full network of distances and no slack. We display the characteristics of the protein (diameter, length n) and the corresponding domain size. We calculated the RMSD between the structure found and the known 3D structure: we show the minimum, the maximum and the average RMSD, plus the standard deviation obtained on 20 independent runs. We also display the average time in seconds on a 1.5GHz pentium. smpl-5 and smpl-8 are partial proteins arbitrarily cut after 50 (resp. 60) residues, showing that the program also works on fragments.

On the full Graph (Table 1) – First we focus on the case where all the exact distances between C_α are provided (the full graph with no slack). With the full network of exact distances, we have an easy distance geometry problem from which the solution could be retrieved in linear time [DW02]. What we intend to prove is the empirical correctness of our algorithm. We say that a solution matches the structure when its RMSD with the known structure is the same order of magnitude than the resolution. Table 1 shows that all the solutions produced match the true structure and that the solution is reached very fast. It is worth noting that although with the full graph of distances the structure solution is unique, its projection to low dimension is not. In practice we observe that the subspace of potential solutions becomes large enough to be explored with little backtracking.

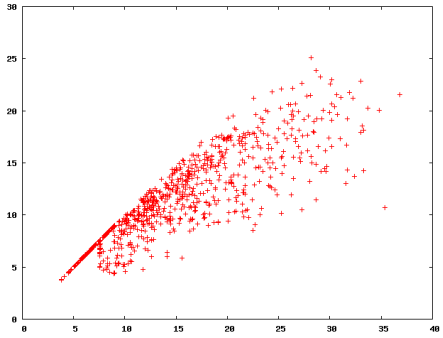
On different reductions to the full graph (Table 2) – Different reductions are applied to the full graph of constraints and we observe the reaction of the solver. We separately test the robustness of the solver toward these different relaxations to the constraints. The network being relaxed, then the number of solutions is larger but they are paradoxically more difficult to retrieve: the solver is more likely to backtrack, as the filters (constraint propagation) do not reduce enough the domains. This can be seen for instance with the rising number of timeouts while we increase the slack (Table 2, smpl-7). We observe that the method is robust to slack (see smpl-7) and sparseness (see smpl-2). A RMSD up to 6Å usually shows a correct dispatching of the backbone into space. With only short range distance, there is a propensity for gathering in the center (as

scope (Å)	slack (%)	keep (%)	RMSD min : **avg** : max ± **stdev**	TIME **avg**
graph density (smpl-2)				
0 .. ∞	± 0	100	1.60 : **1.74** : 1.93 ± **0.09**	**2.30**
0 .. ∞	± 0	10	4.36 : **5.00** : 6.05 ± **0.53**	**140.00** (7)
sensitivity to slack (smpl-7)				
0 .. ∞	± 0	100	1.58 : **1.73** : 1.83 ± **0.07**	**4.30**
0 .. ∞	± 10	100	2.75 : **3.11** : 3.79 ± **0.25**	**7.42** (1)
0 .. ∞	± 20	100	3.72 : **4.69** : 6.23 ± **0.68**	**21.33** (2)
0 .. ∞	± 20	50	4.06 : **5.09** : 6.26 ± **0.66**	**171.44** (2)
0 .. ∞	± 30	100	4.92 : **6.28** : 8.37 ± **1.07**	**45.53** (3)
0 .. ∞	± 40	100	5.18 : **7.54** : 9.50 ± **1.42**	**92.33** (5)
long range influence (smpl-4)				
0 .. 9	± 10	100	7.42 : **8.69** : 10.74 ± **0.78**	**345.33** (8)
0 .. 9 : 30 .. ∞	± 10	100	6.52 : **7.81** : 8.79 ± **0.79**	**344.00** (10)
0 .. 9 : 25 .. ∞	± 10	100	4.69 : **6.02** : 7.51 ± **0.79**	**306.36** (9)

Table 2. Behavior of the model on different types of reductions and relaxations on the full graph of distances. The **scope** is a cutoff on the distance, either low or/and high: 0 .. 9 for instance means that we only keep constraints smaller than 9Å. We indicate the **slack** added and subtracted to the constraints, and the percentage of constraints we **keep** on the full graph. We display the number of timeouts (in parenthesis after the time) ; the time and RMSD measurements were taken on the remainder.

observed in the cuboid model), and search needs some counterbalance: on smpl-4, we notice the beneficial effect of adding long range distances.

Using a heuristic based on shortest paths (Table 3) – We just observed that long range distances improve the quality of the solution, though the constraints coming from NMR have the peculiarity to be only short range distances. Thus we try to infer long range constraints from the topology of the network itself: in the weighted graph of distances (the weight being the average distance $(\delta_{min} + \delta_{max})/2$), the shortest path between any pair of unconnected C_α is computed via Dijkstra's algorithm [CLRS01]. Although this measure may be quite larger than the actual distance, it may be used as a very relaxed constraint (we display the correlation between the shortest path and the actual distance on smpl-1 in Table 3). We used this measure as a heuristic by adding, for a given shortest



Distance in Å between 2 atoms in function of their shortest path in the constraint network (smpl-1 – scope = 9Å– keep = 50%).

scope (Å)	slack (%)	keep (%)	RMSD min : **avg** : max ± **stdev**	TIME **avg**
0 .. 9	± 10	50	5.36 : **7.02** : 8.30 ± **0.70**	**5.35**
0 .. 9 (*)	± 10	50	4.31 : **5.19** : 6.17 ± **0.47**	**5.15**

Table 3. Using a heuristic based on the shortest path (smpl-1). In the second test (marked by a star), we added, for a given shortest path length, the minimum value that we statistically observed in the correlation graph (see above).

path length, the minimum value that we statistically observed in the correlation graph. The benefits of the heuristic are evident (see Table 3) and invite us to explore this approach of adding loose distance constraints, given the robustness of our new model with regard to these constraints.

4 Conclusion and further work

Dealing with an approximate model has advantages and disadvantages. The model might be difficult to handle with precision but it can be robust to slack in the constraints, and projecting NMR constraints onto the backbone results a looser constraint network. The modeling of a problem is far from being canonical, thus we focused here on the modeling itself. In this paper we started from the model we initiated in [KB99] and showed how we could improve both its precision (thus a better manipulation for the design of heuristics) and its accuracy when submitted to loose constraints (short range distances, sparseness and slack). These improvements allow us to use the model at the backbone level, whereas the usual methods for protein structure prediction would hardly cope with loose distances, due to the propagation of errors when the distances have some slack. We found a promising model, which is robust to looseness in the constraints, and

can retrieve solutions with little backtracking. Our ongoing work focuses on the design of efficient heuristics in that model, with a smarter use of the features of the constraint network to infer long range distances.

References

[Bes94] Christian Bessière. Arc-consistency and arc-consistency again. *Artif. Intell.*, 65(1):179–190, 1994.

[BKL96] Bonnie Berger, Jon Kleinberg, and Tom Leighton. Reconstructing a three-dimensional model with arbitrary errors. pages 449–458, 1996.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

[DW02] Qunfeng Dong and Zhijun Wu. A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *J. of Global Optimization*, 22(1-4):365–375, 2002.

[GMW97] P. Güntert, C. Mumenthaler, and K. Wüthrich. Torsion angle dynamics for NMR structure calculation with the new program DYANA. *J. Mol. Biol.*, 273:283–298, 1997.

[Hen92] Bruce Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65–84, 1992.

[HL92] D.A. Hinds and M. Levitt. A lattice model for protein structure prediction at low resolution. volume 89, pages 2536–2540., 1992.

[KB99] Ludwig Krippahl and Pedro Barahona. Applying constraint programming to protein structure determination. In *Principles and Practice of Constraint Programming*, pages 289–302, 1999.

[KB02] Ludwig Krippahl and Pedro Barahona. Psico: Solving protein structures with constraint programming and optimization. *Constraints*, 7(3-4):317–331, 2002.

[KB03] Ludwig Krippahl and Pedro Barahona. Propagating n-ary rigid-body constraints. In Francesca Rossi, editor, *CP*, volume 2833 of *Lecture Notes in Computer Science*, pages 452–465. Springer, 2003.

[KRB99] S. KT, B. Ruczinski, and I. Baker. Ab initio protein structure prediction of casp iii targets using rosetta, 1999.

[Mac75] Alan K. Mackworth. Consistency in networks of relations. Technical report, Vancouver, BC, Canada, Canada, 1975.

[MW99] Jorge J. Moré and Zhijun Wu. Distance geometry optimization for protein structures. *J. of Global Optimization*, 15(3):219–234, 1999.

[Wil02] Sebastian Will. Constraint-based hydrophobic core construction for protein structure prediction in the face-centered-cubic lattice. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing 2002 (PSB 2002)*, pages 661–672, Singapore, 2002. World Scientific Publishing Co. Pte. Ltd.