Maxent-Stress Optimization of 3D Biomolecular Models*†

Michael Wegner[1], Oskar Taubert[2], Alexander Schug[3], and Henning Meyerhenke[4]

1 **Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany**
2 **Department of Physics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany**
3 **Steinbuch Centre for Computing, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany**
4 **Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany**

Abstract

Knowing a biomolecule's structure is inherently linked to and a prerequisite for any detailed understanding of its function. Significant effort has gone into developing technologies for structural characterization. These technologies do not directly provide 3D structures; instead they typically yield noisy and erroneous distance information between specific entities such as atoms or residues, which have to be translated into consistent 3D models.

Here we present an approach for this translation process based on maxent-stress optimization. Our new approach extends the original graph drawing method for the new application's specifics by introducing additional constraints and confidence values as well as algorithmic components. Extensive experiments demonstrate that our approach infers structural models (*i. e.,* sensible 3D coordinates for the molecule's atoms) that correspond well to the distance information, can handle noisy and error-prone data, and is considerably faster than established tools. Our results promise to allow domain scientists nearly-interactive structural modeling based on distance constraints.

1998 ACM Subject Classification G.2.2 Graph Theory, G.1.6 Optimization

Keywords and phrases Distance geometry, protein structure determination, 3D graph drawing, maxent-stress optimization

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.70

1 Introduction

Context. Proteins are biomolecular machines that fulfill a large variety of tasks in living systems, be it as reaction catalysts, molecular sensors, immune responses, or driving muscular activity [40]. Knowing a protein's 3D structure is a requirement for any detailed understanding of its function, and functional or structural disorder can lead to disease. Structure resolution techniques have made strong progress in recent years: biomolecules that were inaccessible a decade ago can now be structurally resolved, as exemplified by the rapid growth of structural

* The full version of this paper is available at `https://arxiv.org/abs/1706.06805`.
† The work by MW and HM was partially supported by grant ME 3619/3-1 within German Research Foundation (DFG) Priority Programme 1736. AS acknowledges support by the *Helmholtz Impuls- und Vernetzungsfonds* and a Google Research Award. HM and AS acknowledge support by KIT's Young Investigator Network YIN.

databases [31]. The resolution techniques, however, do not directly provide structural information as 3D coordinates. Instead, *e. g.,* X-ray crystallography yields a diffraction pattern which has to be translated into a structural model. Similarly, Nuclear Magnetic Resonance (NMR) techniques measure coupled atomic spins, which can be translated into pairwise distances between specific atoms. These distances are typically incomplete, *i. e.,* not all spatially adjacent atom pairs are detected [37]. Also computational tools, such as co-evolutionary analysis of multiple-sequences alignments of protein [35, 39, 29] or RNA families [11] can provide distance constraints between residues for biomolecular structure prediction. Conceptually, one can understand the information provided by these techniques as incomplete and erroneous parts of the complete distance matrix.

Motivation. Our goal is to provide an efficient and effective method to compute a full structural 3D model from incomplete and/or noisy distances. For this task, physics-based approaches are computationally prohibitive. Molecular dynamics-based approaches require weeks on supercomputers [17] and stochastic global optimization techniques [33] still days on medium-sized clusters. To lower computational costs, one can use more coarse-grained force-fields [30], yet the computations still require hours to days depending on the input size.

For interactive or nearly-interactive work, however, running times of a few seconds would be desirable. This would support a quick back-and-forth between, *e. g.,* assigning NMR chemical shifts to determine pairwise atomic distances and follow-up structural modeling [37]. These structural models allow then, in turn, improved NMR shift assignment and a repetition of the loop. One forgoes describing the detailed physics and finds a near-optimal solution which respects all distance constraints. Exemplary tools which solve this distance geometry problem are DGSOL [26] and DISCO [23]. Challenges for such tools are efficiency, quality, and support for variations such as the ability to deal with noise, error, or distance intervals.

Outline and Contribution. In this paper we transfer a maxent-stress optimization approach from 2D graph drawing [16] to computing a 3D model from (incomplete) distance information. To this end, we exploit the resemblance of the objective functions (see Section 2), and extend the basic model and algorithm shown in Section 2.3 by specifics of the biological application. Our algorithmic adaptations and extensions, as well as details regarding their implementation, are presented in Section 3. Extensive experiments (see Section 4) reveal that our algorithm is significantly faster than other competitive algorithms; at the same time its solution quality is very often better than the results of the best competitor. In particular in our most realistic instances, we outperform our competitors (i) in terms of quality by providing more accurate structural models in (ii) consistently high agreement with reference models and requiring (iii) only about 5–10% of the running time. This stays true even when provided with noisy input data. We further extend our algorithm to support a weighted problem variant that allows to specify how certain a distance interval is and obtain very promising experimental results for this setting as well. To our knowledge, our algorithm is the first to support this new variant.

2 Preliminaries

2.1 Problem Definition

We model a biomolecule as a graph $G = (V, E)$, where the set V of n vertices models the atoms and the set E of m edges their relations. Distance information is given separately for all pairs $\{u, v\} \in S \subseteq V \times V$ by a distance matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$. For this purpose d_{vw} denotes

the distance between vertices v and w – or is set to `nil` if the distance is unknown (for pairs $\notin S$). We are interested in finding an embedding of G into $\mathbb{R}^3$, *i. e.,* 3D coordinates for the vertices, that respects the distances for S. This problem is known as (3D) *distance geometry problem* (DGP). In line with previous work, we account for inexact distance information due to measurement errors by introducing an interval in which the actual distance is contained. This modified DGP is called *interval distance geometry problem* [20]:

▶ **Definition 1** (Interval Distance Geometry Problem (iDGP))**.** Let a simple undirected graph $G = (V, E)$, a distance interval function $d = [l, u]$ with $d : E \to \mathbb{R}^2$, and an integer $k > 0$ be given. Then determine whether there is an embedding $x : V \to \mathbb{R}^k$ such that

$$\forall \{v, w\} \in E : l_{vw} \leq \|x_v - x_w\| \leq u_{vw}, \tag{1}$$

where l_{vw} and u_{vw} are lower and upper bounds for the distance of the edge $\{v, w\}$.

Here and in the following, k equals 3. Then DGP is prefixed by an 'M' for 'molecular'. Note that the (M)DGP is contained in the i(M)DGP by setting the lower and upper bound of each interval to be equal to the actual distance. Saxe [32] showed that deciding whether a valid embedding exists (in the DGP sense) is strongly NP-complete for $k = 1$ and strongly NP-hard for $k > 1$. Interestingly, the problem becomes solvable in polynomial time if all distances are given [6, 9, 12]. Since solving the decision problem (finding a valid embedding) is difficult and even not always possible, we continue by considering the embedding task as an optimization problem, to be solved by heuristics. As a measure of error, one could use the *largest distance mean error* (LDME) defined as: $\text{LDME}(x) = \sqrt{\frac{1}{|E|} \sum_{\{v,w\} \in E} \max(l_{vw} - \|x_v - x_w\|, \|x_v - x_w\| - u_{vw}, 0)^2}$. An embedding x that has an LDME value of 0 is obviously a solution of the iDGP as each distance constraint is met. One could thus minimize the LDME of the embedding found. To be closer to the biophysical application and real-world data, we actually use the *root mean square deviation* (RMSD) to *evaluate* our solutions. The RMSD compares the embedding against a reference structure:

$$\text{RMSD}(x, x') = \min \sqrt{\frac{1}{|V|} \sum_{v \in V} \|x_v - x'_v\|^2}, \tag{2}$$

with x_v and x'_v being the coordinates of the embedding and the reference structure, respectively. The minimum value is over all possible spatial translations and rotations of both superimposed structures. RMSD values < 1.5 Å approach the structure resolution limit of experimental wet-lab techniques (NMR and X-ray) [40]. While error functions that test the capability of the algorithm to match the constraints (such as LDME) can be useful from an optimization point of view, a good value does not necessarily mean that the embedding reproduces the molecular structure. In particular, the algorithm must be able to handle noisy and erroneous data. In the end, the structure must also be physically meaningful. The RMSD addresses this challenge and is a standard measure of (dis)similarity in structural molecular biology by directly assessing the usefulness in a real-life scenario[40]. We will therefore rely on the RMSD to compare algorithms.

2.2 Related Work

There exist many algorithms for solving the MDGP optimization problem. Yet, for most algorithms the required running time is either very high or the solution quality is in the meantime rather low. Also, some algorithms currently do not support iMDGP – which limits

their use in a real-world scenario. Due to space constraints the description of related work focuses on two tools, DGSOL and DISCO, which we use in our experimental comparison as they are publicly available and established in the distance geometry research community – cf. a book on distance geometry edited by Mucherino *et al.* [27] and an even more recent survey by the same authors [20]. For a much broader overview we refer to this book and survey. Mentioned running times (in seconds) are based on experimental data in previous works and are thus not necessarily completely comparable.

Liberti *et al.* [20] found in their survey from 2014 that general-purpose global optimization solvers like Octave's *fsolve* or spatial branch-and-bound techniques are not able to solve the problem for more than 10 vertices in a reasonable amount of time. One reason is that the objective function has a large number of local minima.

Moré and Wu [25] implemented an algorithm called DGSOL[1] that transforms the objective function gradually into a smoother function that approximates the original function and has fewer local minima. The algorithm builds a hierarchy of increasingly smooth functions by iteratively applying a transformation. In a next step, DGSOL employs Newton's method as a local optimization on the smoothest function and then traces this solution back to the original objective function, applying a local optimization on each level. Liberti *et al.* state that the algorithm "has several advantages: it is efficient, effective for small-to-medium-sized instances, and, more importantly, can be naturally extended to solve iMDGP instances" [20, p. 23]. On the downside, on large-scale instances the solution quality decreases (while the running time remains reasonable). An et al. [2, 1] use a different continuation approach which improves the solution quality compared to DGSOL. In their experiments the running time of their algorithm for proteins larger than 1 500 atoms lies between 500 and 1 200s. The double variable neighborhood search with smoothing (DVS) algorithm by Liberti et al. [19] combines the ideas of DGSOL and variable neighborhood search into one algorithm. In their comparison to DGSOL the quality of DVS was significantly better, but the running time two orders of magnitude slower already for small inputs.

Biswas *et al.* [5] proposed the DAFGL algorithm that decomposes the graph into clusters by running the symmetric reverse Cuthill-McKee algorithm on the distance matrix. The subproblems are solved with a semidefinite programming (SDP) formulation. DAFGL is capable of solving the iMDGP. While its solution quality are mostly reasonable, one has to consider that as much as 70% of distances smaller than 6 Å were provided with added noise. Also, larger instances increase the running time rapidly. The two largest molecules (PDB: 1toa and 1mqq, see Table 1) took already 2 654s and 1 683s with only modest to poor solution quality (RMSD: 3.2 Å and 9.8 Å) to solve even though the algorithm makes use of a distributed SDP solver. Leung and Toh [18] proposed the DISCO[2] algorithm that is an advancement to the DAFGL algorithm by Biswas *et al.* [5]. If the problem is small enough, DISCO solves the problem with an SDP approach and refines the obtained solution with regularized gradient descent. Otherwise, DISCO splits the graph into two subgraphs and solves the problem recursively. DISCO uses the symmetric reverse Cuthill-McKee algorithm to cluster the vertices initially. In a second step DISCO tries to minimize the edge cut between different subgraphs by placing a vertex v into the subgraph where most of its neighbors are placed. The algorithm puts some vertices in both subgraphs (overlapping atoms) to later stitch the two embedded subgraphs together. Its authors tested DISCO also in the iMDGP setting (*i. e.,* DISCO supports inexact distances). The results indicate that DISCO is able to

[1] Publicly available at `http://www.mcs.anl.gov/~more/dgsol/` (accessed on April 4, 2017).
[2] Publicly available at `http://www.math.nus.edu.sg/~mattohkc/disco.html` (accessed on April 4, 2017).

compute the structure of proteins with very sparse distance data and high noise in 412s for a protein having 3 672 atoms. Fang and Toh [14] presented some enhancements to DISCO for the iMDGP setting by incorporating knowledge about molecule conformations to improve the robustness of DISCO. Their experiments show that their changes indeed lead to better solutions (about 50–70%) with the cost of increased running times (also about 50–70%).

2.3 MaxEnt-Stress Optimization

We aim at developing an algorithm for iMDGP (and its extension wiMDGP, see Section 3.2) with a significantly lower running time than previous algorithms and with solutions of good quality. Our main idea is to use an objective function proposed by Gansner *et al.* [16] for planar graph drawing, called maxent-stress (short for *maximal entropy stress*). As the name suggests, it is composed of two parts, a stress part that penalizes deviations from the prescribed distances (with a quadratic penalty, possibly weighted) and an entropy part that penalizes vertices for getting too close to each other (atoms cannot overlap):

$$\min_x \sum_{vw \in S} \omega_{vw}(\|x_v - x_w\| - d_{vw})^2 \; - \alpha H(x), \tag{3}$$

where $H(x) = -\operatorname{sgn}(q) \sum_{vw \notin S} \|x_v - x_w\|^{-q}, q > -2$, is the entropy term, ω_{vw} a weighting factor for edge $\{v, w\}$, $\alpha \geq 0$ a user-defined control parameter, and $\operatorname{sgn}(q)$ the signum function with the special case that $sgn(0) = 1$.

To minimize function (3), Gansner *et al.* [16] derive a solution from successively solving Laplacian linear systems of the form $Lx = b$ for x. A noteworthy feature of this successive iteration towards a local minimum is that the solution of the current iteration depends on the solution of the previous iteration, since the current right-hand side is computed from the solution in the previous iteration. Note that the computation of distances between vertex pairs not in S is not required for function (3). Instead, vertex pairs not in S are related to each other via the entropy term, which enters the right-hand side, too. If the parameter q is set to be smaller than zero, the entropy term of vertex u acts as a sum of attractive forces on u. Conversely, the term acts as a sum of repulsive forces if q is larger than zero.

The Gansner *et al.* algorithm typically needs several iterations to converge. In this process the entropy weighting factor α has a strong influence in the maxent-stress model: a high value will cause the vertices to expand into space indefinitely while a low value will cause no entropy influence. The maxent-stress algorithm therefore starts with $\alpha = 1$ and gradually reduces this value to $\alpha = 0.008$ with a rate of 0.3. For each value of α, a maximum of 50 linear system solves are performed and Gansner *et al.* set q to 0 except when the graph has more than 30% degree-1 vertices (then $q \leftarrow 0.8$). Note that in this entropy context they assume $\|x\|^0 = \ln \|x\|$. If the relative difference $\|x' - x\|/\|x\|$ between two successive solutions x and x' is below 0.001, the algorithm is terminated.

More implementation details (*e. g.,* the approximation of the entropy term in case of $|S| \in \mathcal{O}(n)$) can be found in Section 3.3.

3 New Algorithm and its Implementation

Now we describe the adaptations made to the generic maxent-stress algorithm in order to deal with iMDGP and an extension called wiMDGP. For more technical details we refer the interested reader to our code[3].

[3] `https://algohub.iti.kit.edu/parco/NetworKit/NetworKit-MaxentStress`, main source folder `networkit/cpp/viz`. An updated standalone version is planned to be published in the future.

3.1 Adapting the Maxent-stress Algorithm for iMDGP

Recall that for iMDGP we are given a graph $G = (V, E)$ and the distance intervals $d = [l, u] : E \to \mathbb{R}^2_{\geq 0}$. We then want to find an embedding $x : V \to \mathbb{R}^3$ that respects the intervals as well as possible. Note that, in line with previous work, we assume the set S of known distances to be equal to E here. We also assume the edges in E to be unweighted.

As Gansner *et al.*'s maxent-stress algorithm [16] cannot cope with intervals, we solve the iMDGP by first running our implementation of the maxent-stress algorithm with an adapted distance $d' : E \to \mathbb{R}$ that is defined by $d'_{vw} \coloneqq (l_{vw} + u_{vw})/2$. One might expect that, in the resulting embedding, the distances should be roughly in the middle of their interval. This would already be a valid solution to the iMDGP. However, our preliminary experiments show that the output of the maxent-stress algorithm still violates distance constraints even on smaller graphs if called this way. We therefore apply local optimizations to the layout computed by the maxent-stress algorithm. One optimization is based on simulated annealing, while the other is a simple local optimization algorithm.

Optimization of the Embedding with Simulated Annealing. Simulated annealing (SA) is a well-known metaheuristic that can escape local optima by probabilistically accepting neighbor solutions that are worse than the current one, see *e. g.* Talbi [38].

Our SA algorithm is sketched as Algorithm 1 in the full version of this paper [41]. The SA metaheuristic is often especially powerful if the initial solution is randomly chosen and can then jump between local minima. In our setting we receive an embedding from the maxent-stress algorithm as input; its global structure should be already quite good and only some of the given distances are not in their desired intervals. Therefore, our SA algorithm is only used to overcome rather narrow local minima instead of jumping to a completely different solution. The constants in Algorithm 1 have been manually chosen in informal experiments. The outermost loop breaks after a certain number of unsuccessful improvement attempts or if the SA temperature is really low. The second loop iterates until an equilibrium w. r. t. the current temperature is reached – here controlled by the number of iterations and modifications. As we do not want to get completely different solutions for reasons mentioned above, we choose a low start temperature and decrease it rather quickly.

Within the innermost loop a new neighbor solution is computed. In fact, we use parallelism here to reduce the running time of the algorithm. While this can lead to some data races if the position of a vertex is altered by more than one thread, we did not experience any significant decrease in quality. The number of total iterations, steps with no improvement and the number of modifications are all chosen rather small to keep the running time low.

The main ingredients of the innermost loop are

(i) the local error criterion,
(ii) the local optimizer that computes a neighbor solution, and
(iii) the acceptance function.

We define $\text{error}_{vw}(x)$ as $\text{error}_{vw}(x) \coloneqq \max\{l_{vw} - \|x_v - x_w\|, \|x_v - x_w\| - u_{vw}, 0\}^2$. and the local error of an edge $\{v, w\}$ as:

$$\text{localError}(\{v, w\}, x) \coloneqq \text{error}_{vw}(x) + \sum_{u \in N(v) \setminus \{w\}} \text{error}_{vw}(x) + \sum_{u \in N(w) \setminus \{v\}} \text{error}_{uw}(x),$$

where $N(v)$ denotes the neighborhood (*i. e.* the set of incident vertices) of v.

In each iteration a new neighbor solution is computed for an edge $\{v, w\}$. To this end, we apply a force-based approach that takes the edge lengths to their neighbors and the edge length of $\{v, w\}$ into account. The idea is to model the local system similarly to the spring

embedder model [13] and the force-directed algorithm by Fruchterman and Reingold [15]. The difference to those algorithms is that we have to deal with an interval for the length of an edge. In our local force optimization step, we only change the positions of v and w while keeping adjacent vertices fixed. We say an edge $\{v, w\}$ is *violating* its distance constraint if the $\text{error}_{vw}(x)$ is larger than 10^{-9} (and not exactly 0 due to numerical reasons). In our spring model only the violating edges account for a repulsive or attractive force, while the other edges do not take part in the force calculation. In our model each spring has its equilibrium state in an interval that corresponds to the interval of the respective edge it models.

For an optimization on edge $\{v, w\}$, the forces acting on v and w are a combination of attractive and repulsive forces: $f(v) \coloneqq f_{rep}(v) + f_{attr}(v)$ and $f(w) \coloneqq f_{rep}(w) + f_{attr}(w)$. The repulsive and attractive forces for a vertex v are defined as $f_{rep}(v) \coloneqq \sum_{w \in N_{rep}(v)} (x_v - x_w) \cdot \frac{l_{wv}^2}{\|x_w - x_v\|^2}$ and $f_{attr}(v) \coloneqq \sum_{w \in N_{attr}(v)} (x_w - x_v) \cdot \frac{u_{wv}^2}{\|x_w - x_v\|^2}$, where $N_{rep}(v) \subseteq N(v)$ is the set of neighbors of v that are too close to v (*i. e.,* the edge is shorter than its lower bound) and $N_{attr}(v) \subseteq N(v)$ is the set of neighbors of v that are too far away (*i. e.,* the edge is longer than its upper bound). Finally, the acceptance function always accepts improving changes. Error-increasing changes are probabilistically accepted according to the Boltzmann distribution based on the local error (as in many cases [38]).

A Simple Local Optimization Algorithm. In addition to our SA optimization algorithm, we propose another simple local optimization algorithm. During one iteration the algorithm sorts the edges by their error (*i. e.,* the deviation from the edge's given distance interval) in descending order. For an edge $\{v, w\}$ having a length that is not in the given distance interval, the algorithm either prolongates or shortens the edge length such that it is exactly as long as the upper or lower bound given by the distance interval, respectively. We only accept the change if we reduce the local edge error. If we change the length of an edge $\{v, w\}$, we lock the other incident edges of v and w for the remainder of the current iteration to prevent an oscillating effect. We perform a maximum of 50 iterations or less if there is no improvement between two successive iterations. Pseudocode of the method is shown as Algorithm 2 in the full version of this paper [41].

3.2 Intervals with Confidence Values: wiMDGP

Some distances can be measured more accurately than others in common biomolecular experimental methods. To account for this, we add a confidence to each interval. Such a confidence states how certain it is that the actual distance is contained in this interval, leading us to the following problem definition:

▶ **Definition 2** (Weighted Interval Distance Geometry (Optimization) Problem (wiDGP))**.** Let a simple undirected graph $G = (V, E)$, a distance interval function $d = [l, u]$, a confidence function $p : E \to \mathbb{R}$, and an integer $k > 0$ be given. Then minimize the following function:

$$\sum_{\{v,w\} \in E} \omega_{vw} \cdot \text{error}_{\{v,w\}}(x), \tag{4}$$

where the weight ω_{vw} depends on the edge's confidence value c_{vw}.

In order to support wiMDGP, we adapt the maxent-stress algorithm as well as the other two optimization algorithms. For wiMDGP we can use the weights ω_{vw} from Eq. (3), the maxent-stress optimization problem, as a penalty that increases the error of an edge if the confidence that the distance lies in the interval is high. After some manual parameter

tuning, we have chosen the following function to define the weighting factors: $\omega_{vw} := 1 + 5 \exp^{-5(1-c_{vw})}$, where c_{vw} denotes the confidence for edge $\{v, w\}$. This way, the weight of an edge is roughly in the interval $[1, 6]$ and increases rapidly between 0.7 and 1. A confidence between 0 and 0.6 only tells us that we cannot be very certain about the distance and thus, the error term should not vary too much. For higher confidence values, we can be quite certain that the distance interval is correct, so we need to penalize the errors of such edges significantly higher. Choosing a larger interval for the weights turned out not to be beneficial in terms of solution quality in preliminary experiments.

Both our SA and simple local optimization algorithm use an adapted error function for an edge that includes the weighting function above. In our simple local optimizer, we additionally change the sorting of the edges such that edges with higher confidence are considered first. Confidence ties are broken by choosing the edge with larger error first.

3.3 Implementation Details

Initial layout. For computing the initial layout, we implemented three algorithms. In addition to PivotMDS [7], which was used by Gansner *et al.* [16], these are two random vertex placement algorithms. The first one is a very simple method and places the coordinates randomly in a k-dimensional hypercube with predefined side length.

The second one does include some of the distance information provided by the input. Given an edge $\{v, w\}$ and the coordinates of vertex v, we place w at the boundary of a k-dimensional hypersphere with radius d_{vw} and centered at v. Some more details can be found in the full version [41].

Finally, PivotMDS is an approximation algorithm for multidimensional scaling that is based on sampling; for a detailed description the reader is referred to Brandes and Pich [7].

Preliminary experiments indicated that PivotMDS and the random hypersphere approach fare similarly well. Since PivotMDS turned out to be more robust in terms of solution quality when applied to protein instances, we use it in all the following experiments. Its slower speed is more than outweighed by the more expensive maxent-stress algorithm.

Approximating the entropy term. The entropy term in Eq. (3) iterates over all elements not in S. As the set S is usually sparse, this computation would thus require quadratic running time. Thus, it is important to approximate the distances required for the entropy calculations. We implemented both the well-known Barnes-Hut approximation (also used by Gansner *et al.*) as well as well-separated pair decomposition (WSPD) [8].

Additionally, we evaluate the entropy lazily: instead of computing the entropy term in each iteration, we recompute it only when the function $\lfloor 5 \log i \rfloor$ changes, where i is the iteration number. We expect the entropy term to significantly change more frequently at the beginning of a new iteration; thus, we use a function that causes the algorithm to recompute the entropy more often at lower iteration numbers. Lipp *et al.* [21] use the same function for reducing the running time of their WSPD algorithm. This "lazy" computation of the entropy term significantly reduces the running time while the quality does not deteriorate much.

Solving the Laplacian linear systems. Recall that Gansner *et al.* derived an iteration of subsequent Laplacian linear systems for optimizing maxent-stress. They use the conjugate gradient method (CG) as a Laplacian solver in their implementation. The conjugate gradient method has superlinear time complexity. That is why we use lean algebraic multigrid (LAMG) instead, a fast solver proposed by Livne and Brandt [22] with linear empirical running time. We use our C++ implementation of LAMG; it is available in NetworKit and has been used

Table 1 Proteins for distance geometry benchmarks and their basic properties [4]. Listed are the protein data bank (PDB) code [31], and the size in atoms (vertices), amino acid residues, the number of edges equivalent to covalent bonds (= bonds edges) and the number of edges with atoms closer to each other than 5Å without being a covalent bond (= contact edges).

Protein	# atom/ vertices	# residues	bond edges	contact edges
1ptq	402	50	412	3987
1lfb	641	78	654	6320
1gpv	735	87	696	7208
1f39	767	101	788	7621
1ax8	1003	130	1016	10527
1rgs	2015	264	2053	20731
1toa	2138	277	2181	23168
1kdh	2846	356	2904	30655
1bpm	3671	481	3744	41027
1mqq	5510	675	5665	62396

for other Laplacian graph problems before [3]. An alternative multilevel approach for solving the linear systems exists [24], but is harder to adapt to the present scenario.

Optimization Workflow. We combine our two optimization algorithms into one workflow: the solution found by the SA algorithm can often be further improved by a subsequent run of our simple local optimization algorithm. Sometimes it happens that the SA algorithm only finds a slightly worse solution compared to the maxent-stress algorithm. In this case we ignore the SA solution and only run the simple optimization algorithm.

4 Experiments

In this section we present a representative subset of our experiments and their results. To this end, we implemented our algorithm in C++ based on NetworKit [36], an open-source toolkit for graph algorithms and in particular interactive large-scale network analysis. We call our algorithm MOBi (for **M**axent-stress **O**ptimization of **Bi**omolecular models) and compare it with DGSOL [25] (C/Fortran code) and DISCO [18] (compiled Matlab code), two of the very few publicly available established tools that can handle inexact inputs with intervals. Experimental settings are further detailed in the full version of this paper [41].

4.1 Instances

To test the accuracy and efficiency of our approach in a real-world setting, we work on 10 proteins of different sizes [4] taken from the protein data bank (PDB) [31], see Table 1. These proteins range from small globular proteins with 50 amino acids to large proteins with about 700 amino acids. We only consider *ATOM* entries in the file, which provide atomic coordinates. Also, we only work on the first chain in the case of multiple protein chains. Based on the coordinates of a protein, we can construct an instance for the various distance geometry problems. For each experiment we actually create three instances per protein (*i. e.,* different contact distance information) to eliminate effects of particularly good/bad sets of input data. Also, each instance (*i. e.,* same contact distance information) is re-run three times to eliminate stochasticity effects. The displayed data per protein are averaged over these three instances and three respective runs.

We use a percentage p of all atom distances $\{v, w\} \in \binom{V}{2}$ for which $d_{vw} = \|x_v - x_w\|$ is below a cut-off distance of 5 Å, where x denotes the coordinates from the protein file. We use 5 Å because this approximates the distance that can be determined by NMR experiments [42, 37] and since it is a typical cutoff in determining so-called contact maps (*i. e.* binary matrices that store only adjacencies whose length is below the cutoff) [34, 28]. To construct an instance for iMDGP, we introduce the interval $[d_{vw} - \underline{\epsilon}, d_{vw} + \overline{\epsilon}]$, where $\underline{\epsilon}, \overline{\epsilon} = d_{vw} \cdot \mathcal{N}(0, \sigma^2)$ and σ is the standard deviation. We denote instances created this way as *normal*-iMDGP instances.

In contrast, the *bonds*-iMDGP more closely reflects standard NMR experiments by taking protein biochemistry into account. As all covalent bonds of a protein are known, instances can be assumed to have full knowledge of exact distances for these bonds: Chemically, the length of covalent bonds fluctuates very little, hence the interval for these bonds edges $\{v, w\}$ consists of a single distance $[d_{vw}, d_{vw}]$. In addition to the distance information of the bonds, we add more distances the same way as for constructing a *normal*-iMDGP instance.

4.2 Results & Discussion

To quantify the structure determination quality of the different approaches, we compare them by RMSD. For the *normal*-iDGP test instances, we observe low RMSD for MOBi and DISCO, while DGSOL performs significantly worse (cf. Tables 2 and 3 as well as Table 4 in Appendix A.3 of the full version [41]. As expected, the RMSD gets higher if less distance information is provided (the instances in Tables 2 and 3 provide only 50% and 30% of the contact edges, respectively, while the instances in Table 4 provide 70%). The RMSD values do not increase, as one might expect *a priori*, necessarily with instance size (number of atoms/ vertices). Instead, these instances have more edges, which might make the embedding computationally more demanding but also of good quality. Indeed, MOBi and DISCO yield good embeddings regardless of system size. DGSOL performs worse for larger systems. Also, MOBi is more consistent than DISCO and performs best in nearly all instances, in particular for those with less information (cf. instances 1gpv and 1rgs in Table 3). Similarly, the running times of MOBi are about an order of magnitude faster than DGSOL and DISCO, with DGSOL being slightly faster than DISCO. There is an overall trend of increased running times with system size, but some instances seem particular hard to compute (*e. g.* instances 1gpv, 1rgs, and 1toa in Table 3). Gaussian noise on the intervals does not significantly alter the results: given a relatively high amount of distance information, MOBi and DISCO produce embeddings of very high quality with RMSD < 1 Å (see Tables 4, 5 and 6 in Appendix A.3 in the full version of this paper [41]. It should be noted, though, that in Table 5 DISCO yields the majority of best results in terms of solution quality – but MOBi is usually not far behind.

For the *bonds*-iMDGP instances, we display the results only for MOBi (DISCO and DGSOL show the same respective trends as above) as a heatmap in Figure 1. For very low amounts of additional distance information (1–2%) in addition to the bonds, MOBi is unable to provide high-quality embeddings as displayed by RMSD values > 5 Å. When provided as little as 8–12% distance information in addition to the bonds, the structure determination quality becomes below 3 Å RMSD, *i. e.,* it approaches the wet-lab resolution. Interestingly, the amount of Gaussian noise does not strongly influence the embedding quality. Exemplary structure embeddings are displayed in the full version of this paper [41]. While the overall quality appears good here as well, one can see that most structural errors are found at the surface, where fewer edges are available.

The experimental results for wiMDGP are shown in Table 7 in the full version [41]. As these instances cannot be directly compared against other cases, we merely report that MOBi

Table 2 Performance results on 50% $\sigma = 0.1$ *normal*-iDGP instances. Best results in bold font.

	RMSD / Å			time / s		
Protein	MOBi	DGSOL	DISCO	MOBi	DGSOL	DISCO
1ptq	0.46	8.05	**0.45**	**1.56**	9.85	13.70
1lfb	**0.87**	10.51	1.00	**2.73**	22.99	25.29
1gpv	**0.68**	14.35	0.91	**7.51**	120.35	128.54
1f39	**0.53**	16.45	0.69	**5.55**	70.22	70.80
1ax8	**0.51**	12.23	0.55	**3.91**	39.42	48.49
1rgs	**0.60**	17.56	1.05	**7.10**	112.49	155.52
1kdh	**0.88**	19.41	1.06	**8.64**	173.70	186.00
1toa	**0.48**	23.72	0.86	**14.14**	181.96	311.92
1bpm	**0.48**	21.73	0.50	**12.94**	221.18	264.02
1mqq	**0.34**	23.56	0.40	**18.22**	381.65	519.58

Table 3 Performance results on 30% $\sigma = 0.1$ *normal*-iDGP instances. Best results in bold font.

	RMSD / Å			time / s		
Protein	MOBi	DGSOL	DISCO	MOBi	DGSOL	DISCO
1ptq	**1.05**	9.29	1.09	**1.38**	12.39	13.30
1lfb	**1.50**	11.68	1.53	**2.79**	26.76	22.86
1gpv	**1.29**	16.29	3.48	**6.64**	112.08	114.63
1f39	**1.06**	17.27	1.74	**5.18**	82.75	78.07
1ax8	**1.07**	12.95	1.34	**3.85**	47.51	44.43
1rgs	**1.37**	18.03	5.54	**40.33**	120.89	120.59
1toa	**1.00**	23.47	1.28	**13.10**	204.94	325.64
1kdh	**1.46**	20.33	1.51	**8.28**	174.67	169.41
1bpm	**0.93**	22.08	1.37	**11.49**	237.23	255.87
1mqq	**0.82**	23.45	0.96	**16.36**	216.39	529.09

produces high quality solutions, typically with RMSD < 1.0 Å and comparable running times to the other instances with MOBi. To truly assess the use of this wiMDGP implementation on real-world data, one would have to work on curated wet-lab experimental data [42, 40] or co-evolutionary signals [35]. This is clearly outside the scope of this paper.

Overall, in particular for limited and noisy distance information, MOBi provides consistently embeddings with higher quality and does so at significantly lower running times than both DISCO and DGSOL. On average (geometric means over quotients for each of the Tables 2 to 6), MOBi is between 13x and 20x faster than DISCO. At the same time its RMSD values are on average 17% to 41% better – except for Table 5, where DISCO is 12% better. DGSOL is not competitive in terms of solution quality and also 6x to 13x slower.

For *normal*-iDGP instances and very high amounts of distance information, the MOBi embeddings provide RMSD < 1 Å, which is below the typical resolution of NMR or X-ray [40]; even providing only $p = 30\%$ leads to high quality embeddings. Both MOBi and DISCO perform considerably better than older algorithms such as[5], where some RMSD > 5 Å were reported. In the more realistic scenario of *bonds*-iMDGP instances with all bond edges provided, only few contact edges (8–12%) can already lead to high quality embeddings with MOBi. Thus, we are confident that our algorithm will lead to improved interpretation of wet-lab experiments in particular in cases with sparse data, such as sparse NMR experiments.

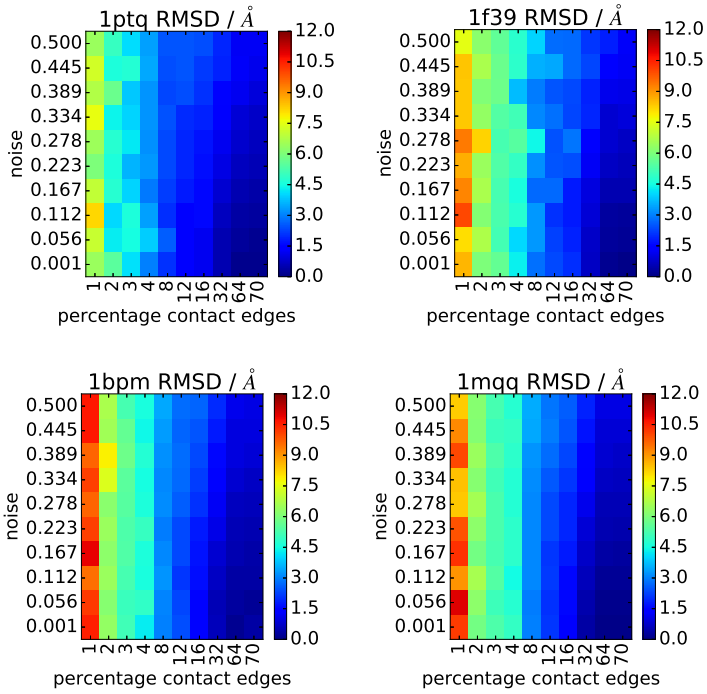

Figure 1 Quality results for bonds-iDGP instances. The horizontal axis shows the amount of contact edges in addition to bond edges provided, the vertical axis varies the σ of Gaussian noise.

5 Conclusions

This paper provides a significant step towards nearly-interactive protein structure determination. To this end, we implemented the maxent-stress algorithm [16] and incorporated first of all a faster Laplacian solver. Based on this implementation we extended the graph drawing algorithm to handle distance geometry problems where the distances are either exact or come in the form of intervals, optionally with some confidence. Comparing our algorithm with two publicly available competitors shows that we are able to significantly outperform both of them in terms of running time, while usually providing embeddings with higher quality. For the more realistic bonds-iDGP instances, our algorithm is able to compute high quality protein structures with limited and noisy information. Most errors can be found at the surface of the proteins, where only few edges can guide the optimization process.

While some related work can provide even higher solution quality (*e. g.* [14]), it can only do so at the expense of an enormous increase in running time. The strength of our work is the combination of low running time, good and consistent solution quality, and genericity.

The evaluation of our algorithm on real-world instances whose distance matrices are derived from chemical bonds and real NMR experiments is ongoing and shows very promising results, too. In the future it also seems promising to use our algorithm for bootstrapping more sophisticated and thus more expensive algorithms for protein structure determination (such as refining the resulting structures in physics-based force fields similar to [35, 10]). Moreover, further improvements of the resulting structures could be achieved by re-weighting edges by their density; such an approach would consider surfaces more strongly.

Acknowledgments. The authors thank Michael Kovermann (University of Konstanz) for fruitful discussions.

References

1 Le Thi Hoai An. Solving large scale molecular distance geometry problems by a smoothing technique via the gaussian transform and d.c. programming. *Journal of Global Optimization*, 27(4):375–397, 2003. doi:10.1023/a:1026016804633.

2 Le Thi Hoai An and Pham Dinh Tao. Large-scale molecular optimization from distance matrices by a d.c. optimization approach. *SIAM Journal on Optimization*, 14(1):77–114, jan 2003. doi:10.1137/s1052623498342794.

3 Elisabetta Bergamini, Michael Wegner, Dimitar Lukarski, and Henning Meyerhenke. Estimating current-flow closeness centrality with a multigrid laplacian solver. In *Proc. 7th SIAM Workshop on Combinatorial Scientific Computing, CSC 2016*, pages 1–12. SIAM, 2016. doi:10.1137/1.9781611974690.ch1.

4 Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, Jan 2000. doi:10.1093/nar/28.1.235.

5 Pratik Biswas, Kim-Chuan Toh, and Yinyu Ye. A distributed SDP approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *SIAM Journal on Scientific Computing*, 30(3):1251–1277, jan 2008. doi:10.1137/05062754x.

6 Leonard M. Blumenthal. *Theory and Applications of Distance Geometry*, volume 347. Oxford, 1953.

7 Ulrik Brandes and Christian Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, pages 42–53. Springer Science + Business Media, 2007. doi:10.1007/978-3-540-70904-6_6.

8 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, jan 1995. doi:10.1145/200836.200853.

9 Gordon M. Crippen, Timothy F. Havel, et al. *Distance Geometry and Molecular Conformation*, volume 74. Research Studies Press Taunton, UK, 1988.

10 Angel E. Dago, Alexander Schug, Andrea Procaccini, James A. Hoch, Martin Weigt, and Hendrik Szurmant. Structural basis of histidine kinase autophosphorylation deduced by integrating genomics, molecular dynamics, and mutagenesis. *Proceedings of the National Academy of Sciences*, 109(26):E1733–E1742, 2012.

11 Eleonora De Leonardis, Benjamin Lutz, Sebastian Ratz, Simona Cocco, Rémi Monasson, Alexander Schug, and Martin Weigt. Direct-coupling analysis of nucleotide coevolution facilitates rna secondary and tertiary structure prediction. *Nucleic acids research*, 43(21):10444–10455, 2015.

12 Qunfeng Dong and Zhijun Wu. A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization*, 22(1/4):365–375, 2002. doi:10.1023/a:1013857218127.

13 Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:146–160, 1984.

14 Xingyuan Fang and Kim-Chuan Toh. Using a distributed SDP approach to solve simulated protein molecular conformation problems. In *Distance Geometry*, pages 351–376. Springer Science + Business Media, nov 2012. doi:10.1007/978-1-4614-5128-0_17.

15 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, Nov 1991. doi:10.1002/spe.4380211102.

16 Emden R. Gansner, Yifan Hu, and Stephen North. A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):927–940, jun 2013. doi:10.1109/tvcg.2012.299.

17 Oliver F. Lange, Nils-Alexander Lakomek, Christophe Farès, Gunnar F. Schröder, Korvin F. A. Walter, Stefan Becker, Jens Meiler, Helmut Grubmüller, Christian Griesinger, and

Bert L. De Groot. Recognition dynamics up to microseconds revealed from an rdc-derived ubiquitin ensemble in solution. *science*, 320(5882):1471–1475, 2008.

18 Ngai-Hang Z. Leung and Kim-Chuan Toh. An SDP-based divide-and-conquer algorithm for large-scale noisy anchor-free graph realization. *SIAM Journal on Scientific Computing*, 31(6):4351–4372, jan 2010. doi:10.1137/080733103.

19 Leo Liberti, Carlile Lavor, Nelson Maculan, and Fabrizio Marinelli. Double variable neighbourhood search with smoothing for the molecular distance geometry problem. *Journal of Global Optimization*, 43(2-3):207–218, aug 2009. doi:10.1007/s10898-007-9218-1.

20 Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, jan 2014. doi:10.1137/120875909.

21 Fabian Lipp, Alexander Wolff, and Johannes Zink. Faster force-directed graph drawing with the well-separated pair decomposition. In *Graph Drawing and Network Visualization – 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*, volume 9411 of *LNCS*, pages 52–59. Springer, 2015. doi:10.1007/978-3-319-27261-0_5.

22 Oren E. Livne and Achi Brandt. Lean algebraic multigrid (LAMG): Fast graph laplacian linear solver. *SIAM Journal on Scientific Computing*, 34(4):B499–B522, Jan 2012. doi:10.1137/110843563.

23 Jeffrey W. Martin, Anthony K. Yan, Chris Bailey-Kellogg, Pei Zhou, and Bruce R. Donald. A geometric arrangement algorithm for structure determination of symmetric protein homooligomers from noes and rdcs. *Journal of Computational Biology*, 18(11):1507–1523, 2011.

24 Henning Meyerhenke, Martin Nöllenburg, and Christian Schulz. Drawing large graphs by multilevel maxent-stress optimization. In *Graph Drawing and Network Visualization – 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*, volume 9411 of *LNCS*, pages 30–43. Springer, 2015. doi:10.1007/978-3-319-27261-0_3.

25 Jorge J. Moré and Zhijun Wu. Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7(3):814–836, aug 1997. doi:10.1137/s1052623495283024.

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

27 Antonio Mucherino, Carlile Lavor, Leo Liberti, and Nelson Maculan, editors. *Distance Geometry: Theory, Methods, and Applications*. Springer Science + Business Media, 2013. doi:10.1007/978-1-4614-5128-0.

28 Jeffre K. Noel, Paul C. Whitford, and Onuchic Jose N. The shadow map: A general contact definition for capturing the dynamics of biomolecular folding and function. *J Phys Chem B*, 116(29):8692–8702, 2013. doi:10.1021/jp300852d.

29 Sergey Ovchinnikov, Hahnbeom Park, Neha Varghese, Po-Ssu Huang, Georgios A. Pavlopoulos, David E. Kim, Hetunandan Kamisetty, Nikos C. Kyrpides, and David Baker. Protein structure determination using metagenome sequence data. *Science*, 355(6322):294–298, 2017.

30 Carol A. Rohl, Charlie E. M. Strauss, Kira M. S. Misura, and David Baker. Protein structure prediction using rosetta. *Methods in enzymology*, 383:66–93, 2004.

31 Peter W. Rose, Andreas Prlić, Chunxiao Bi, Wolfgang F. Bluhm, Cole H. Christie, Shuchismita Dutta, Rachel Kramer Green, David S. Goodsell, John D. Westbrook, Jesse Woo, Jasmine Young, Christine Zardecki, Helen M. Berman, Philip E. Bourne, and Stephen K. Burley. The rcsb protein data bank: views of structural biology for basic and applied research and education. *Nucleic Acids Research*, 43(D1):D345, 2015. doi:10.1093/nar/gku1214.

32 James B. Saxe. *Embeddability of weighted graphs in k-space is strongly NP-hard*. Carnegie-Mellon University, Department of Computer Science, 1980.

33 A. Schug, T. Herges, and W. Wenzel. Reproducible protein folding with the stochastic tunneling method. *Physical review letters*, 91(15):158102, 2003.

34 Alexander Schug and José N. Onuchic. From protein folding to protein function and biomolecular binding by energy landscape theory. *Current opinion in pharmacology*, 10(6):709–714, 2010.

35 Alexander Schug, Martin Weigt, José N. Onuchic, Terence Hwa, and Hendrik Szurmant. High-resolution protein complexes from integrating genomic information with molecular simulation. *Proceedings of the National Academy of Sciences*, 106(52):22124–22129, 2009.

36 Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: A tool suite for large-scale complex network analysis. *Network Science*, 4(4):508–530, Dec 2016.

37 J. B. Stothers. *Carbon-13 NMR Spectroscopy: Organic Chemistry, A Series of Monographs*, volume 24. Elsevier, 2012.

38 El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

39 Guido Uguzzoni, Shalini John Lovis, Francesco Oteri, Alexander Schug, Hendrik Szurmant, and Martin Weigt. Large-scale identification of coevolution signals across homo-oligomeric protein interfaces by direct coupling analysis. *Proceedings of the National Academy of Sciences*, 114(13):E2662–E2671, 2017.

40 D. Voet and J. G. Voet. *Biochemistry, 4th Edition*. John Wiley & Sons, 2010.

41 Michael Wegner, Oskar Taubert, Alexander Schug, and Henning Meyerhenke. Maxent-stress optimization of 3D biomolecular models. *arXiv preprint arXiv:1706.06805*, Jun 2017. URL: `https://arxiv.org/abs/1706.06805`.

42 Kurt Wüthrich. Protein structure determination in solution by nmr spectroscopy. *Journal of Biological Chemistry*, 265(36):22059–22062, 1990.