Diagonally dominant programming in distance geometry

Gustavo Dias* and Leo Liberti**

CNRS LIX, École Polytechnique, 91128 Palaiseau, France
{dias,liberti}@lix.polytechnique.fr

Abstract. Distance geometry is a branch of geometry which puts the concept of distance at its core. The fundamental problem of distance geometry asks to find a realization of a finite, but partially specified, metric space in a Euclidean space of given dimension. An associated problem asks the same question in a Euclidean space of *any* dimension. Both problems have many applications to science and engineering, and many methods have been proposed to solve them. Unless some structure is known about the structure of the instance, it is notoriously difficult to solve these problems computationally, and most methods will either not scale up to useful sizes, or will be unlikely to identify good solutions. We propose a new heuristic algorithm based on a semidefinite programming formulation, a diagonally-dominant inner approximation of Ahmadi and Hall's, a randomized-type rank reduction method of Barvinok's, and a call to a local nonlinear programming solver.

1 Introduction

The main problem studied in this paper is the

> Distance Geometry Problem (DGP). Given an integer $K \geq 1$ and a simple, edge-weighted, undirected graph $G = (V, E, d)$, where $d : E \to \mathbb{R}_+$, verify the existence of a *realization function* $x : V \to \mathbb{R}^K$, i.e. a function such that:
>
> $$\forall \{i, j\} \in E \quad \|x_i - x_j\| = d_{ij}. \tag{1}$$

A recent survey on the DGP with the Euclidean norm is given in [15]. The DGP is **NP**-hard, by reduction from Partition using 2-norms [20]. Three well-known applications are to clock synchronization ($K = 1$), sensor network localization ($K = 2$), and protein conformation ($K = 3$). If distances are Euclidean, the problem is called Euclidean DGP (EDGP) — but, given the preponderance of Euclidean distances in the DGP literature w.r.t. other distances, if the norm is not specified, it is safe to assume the 2-norm is used.

* Financially supported by a CNPq PhD thesis award.
** Partly supported by the ANR "Bip:Bip" project under contract ANR-10-BINF-0003.

A related problem, the Distance Matrix Completion Problem (DMCP), asks whether a partially defined matrix can be completed to a distance matrix. The difference is that while K is part of the input in the DGP, it is part of the output in the DMCP, in that a realization to a Euclidean space of *any* dimension satisfying (1) provides a certificate. When the completion is required to be to a Euclidean distance matrix (EDM), i.e. where distances are given by 2-norms, this problem is called Euclidean DMCP (EDMCP). It is remarkable that, albeit the difference between EDGP and EDMCP is seemingly minor, it is not known whether the EDMCP is in **P** or **NP**-hard (whereas the EDGP is known to be **NP**-hard). The EDMCP is currently thought to be "between the two classes".

In this paper we propose a new heuristic algorithm designed to be accurate yet solve instances of sufficiently large sizes. Our motivation in proposing new heuristics based on Mathematical Programming (MP) formulations is that they can be easily adapted to uncertainty on the distances d_{ij} expressed as intervals, i.e. they can also solve the problem

$$\forall \{i,j\} \in E \quad d_{ij}^L \leq \|x_i - x_j\| \leq d_{ij}^U. \tag{2}$$

This is in contrast to some very fast combinatorial-type algorithms such as the Branch-and-Prune (BP) [14, 13], which are natively limited to solving Eq. (1). In fact, this paper is in support of a study which is auxiliary to the development of the BP algorithm, namely to endow the BP with an ability to treat at least some fraction of the distances being given as intervals, which appears to be the case in practice for protein conformation problems from distances.

Our heuristic has three main ingredients: Diagonally Dominant Programming (DDP), very recently proposed by Ahmadi et al. [18, 1]; a randomized rank-reduction method of Barvinok's [4]; and a call to a general-purpose local Nonlinear Programming (NLP) solver.

DDP is a technique for obtaining a sequence of inner approximating Linear Programs (LP) or Second-Order Cone Programs (SOCP) to semidefinite programming (SDP) formulations. In this paper we only consider the LP variant, since LP solution technology is more advanced than SDP or SOCP. DDP has been proposed in very general terms; its adaptation to (dual) SDP formulations for the DGP yields a valid LP relaxation for the DGP. In this paper, since we are proposing a heuristic method, and need feasible solutions, we apply DDP to a primal SDP formulation of the DGP.

Note that SDP solutions are square symmetric matrices of any rank, whereas a feasible solution of the DGP must have the given rank K. Although there are many rank reduction techniques, most of them do not have guaranteed properties, even in probability, of preserving the feasibility of the solution. In fact, in the case of the DGP, it is exactly this rank constraint which makes the problem hard, so we can hardly hope in an efficient rank reduction technique that works infallibly. We found the next best thing to be a probabilistic rank reduction technique proposed by Barvinok: although it does not exactly preserve feasibility, it gives a probabilistic guarantee that it will place the reduced rank solution fairly close to all of the manifolds $\mathcal{X}_{ij}$ of realizations x satisfying $\|x_i - x_j\| = d_{ij}$ (for

each $\{i,j\} \in E$). At this point, we attempt to achieve feasibility via a single call to a local NLP solver.

Our computational results are preliminary and are simply designed as validation, as this is work-in-progress. We compare the heuristic sketched above to the same, with DDP replaced by SDP. With this limited set-up, we found that the DDP approach exhibits its large-scale potential as the instance sizes increase.

The rest of this paper is organized as follows. We give some technical notation and background in Sect. 1.1. We propose some existing and new SDP formulations of the DGP and EDMCP in Sect. 2. We explain DDP and give a new DDP formulation for the DGP and EDMCP in Sect. 3. We discuss our new DGP heuristic in Sect. 4. We present our preliminary results in Sect. 5. We sketch our roadmap ahead in Sect. 6.

1.1 Relevant background

The EDGP calls for a solution to the set of nonlinear equations

$$\forall \{i,j\} \in E \quad \|x_i - x_j\|_2 = d_{ij}, \tag{3}$$

where $x_i \in \mathbb{R}^K$ for all $i \leq n = |V|$. Usually, the squared version of Eq. (3) is employed, for two reasons: first, since the vast majority of algorithmic implementations employ floating point representations, there is a risk that $\sum_k (x_{ik} - x_{jk})^2 = 0$ might be represented by a tiny negative floating point scalar, resulting in a computational error when extracting the square root. Secondly, as pointed out in [6], the squared EDM $D^2 = (d_{ij}^2)$ has rank at most $K + 2$, a fact which can potentially be exploited. Obviously, solving the squared system yields exactly the same set of solutions as the original system.

Most methods for solving Eq. (3) do not address the original system explicitly, but rather a penalty function:

$$\sum_{\{i,j\} \in E} (\|x_i - x_j\|_2^2 - d_{ij}^2)^2, \tag{4}$$

which has global optimum x^* with value zero if and only if x^* satisfies Eq. (3). This formulation is convenient since most local NLP solvers find it easier to improve the cost of a feasible non-optimal solution, rather than achieving feasibility from an infeasible point. This is relevant since such solvers are often employed to solve EDGP instances. Eq. (4) can be easily adjusted to deal with imprecise distances represented by intervals (see e.g. [17]).

There are several Semidefinite Programming (SDP) relaxations of the EDGP [21, 2, 19], mostly based on linearizing the constraint

$$\forall \{i,j\} \in E \quad \|x_i\|_2^2 + \|x_j\|_2^2 - 2x_i \cdot x_j = d_{ij}^2$$

into

$$\forall \{i,j\} \in E \quad X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \tag{5}$$

and then relaxing the rank constraint $X = xx^\top$ to $X \succeq xx^\top$, which, via the Schur complement, can be written as the semidefinite constraint

$$Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \succeq 0. \tag{6}$$

Such formulations mostly come from the application to sensor networks, for which $K = 2$. In his EE392O course 2003, Y. Ye proposes the objective function:

$$\min \mathsf{tr}(Y), \tag{7}$$

motivated by a probabilistic interpretation of the solution of the SDP. Purely based on (unpublished) empirical observations, we found what is possibly a better objective function (at least for some protein conformation instances), discussed in Sect. 2 below.

Several methods aim to decompose large graphs into rigid components [9, 5, 11], since many rigid graphs can be realized efficiently [7, 16]. Each rigid subgraph realization is then "stiched up" consistently by either global optimization [9] or SDP [5, 11].

One notable limitation of SDP for practical purposes is that current technology still does not allow us to scale up to large-scale instance sizes. More or less, folk-lore says that interior point methods (IPM) for SDP are supposed to work well up to sizes of "around" 1000 variables, i.e. a matrix variable of around 33×33, which is hardly "large-scale". As remarked, a technique which can address this limitation is the very recent DDP [18, 1]. Since all diagonally dominant (DD) matrices are positive semidefinite (PSD), any DDP obtained from an SDP by replacing the PSD constraint with a DD one is an inner approximation of the original SDP. The interesting feature of DDP is that it can be reformulated to an LP, which current technology can solve with up to millions of variables.

Once a solution $\bar{X}$ of an SDP relaxation has been found, the problem of finding another solution of the correct rank, which satisifies $X = xx^\top$ is called *rank reduction*. Possibly the most famous rank reduction algorithm is the Goemans-Williamson algorithm for Max Cut [8]. Other ideas, connected with the concentration of measure phenomenon, have been proposed in [4] in order to find a solution x which is reasonably close, on average and with high probability, from the manifolds $\mathcal{X}_{ij}$ described in Eq. (3).

Although being "reasonably close to a manifold" is certainly no guarantee that a local NLP solver will move the reasonably close point to the manifold itself, there is a good hope of this being the case.

2 SDP formulations for DG

We represent a realization x in matrix form by an $n \times K$ matrix where $n = |V|$, and where each of the n rows is a vector $x_i \in \mathbb{R}^K$ which gives the position of vertex $i \in V$. We discussed a well known SDP for the EDGP in Sect. 1.1, which

we recall here without the objective function, for later reference.

$$\left.\begin{aligned} \forall \{i,j\} \in E \; X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \\ Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \succeq 0. \end{aligned}\right\} \tag{8}$$

2.1 A better objective for protein conformation

The empirical evidence collected by Ye about Eq. (8) with $\min \mathsf{tr}(Y)$ as objective concerns the application of EDGP to the localization of sensor networks. Our own (unpublished and preliminary) computations on protein conformation instances with the above objective were not particularly encouraging. We found relatively better results with a different objective function:

$$\left.\begin{aligned} \min \sum_{\{i,j\} \in E} (X_{ii} + X_{jj} - 2X_{ij}) \\ \forall \{i,j\} \in E \quad X_{ii} + X_{jj} - 2X_{ij} \geq d_{ij}^2 \\ Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \succeq 0. \end{aligned}\right\} \tag{9}$$

Note that Eq. (9) can be trivially derived as the natural SDP relaxation of the the nonconvex NLP:

$$\left.\begin{aligned} \min \sum_{\{i,j\} \in E} \|x_i - x_j\|_2^2 \\ \forall \{i,j\} \in E \quad \|x_i - x_j\|_2^2 \geq d_{ij}^2, \end{aligned}\right\} \tag{10}$$

which is an exact reformulation of Eq. (4) since, if Eq. (3) has a solution x^*, at x^* all of the inequality constraints of Eq. (10) are tight, and therefore the objective cannot be further decreased. Conversely, if there was an x' with lower objective function value, at least one of the constraints would be violated.

For the EDMCP, where the rank is of no importance, we only require that X should be the Gram matrix of a realization x (of any rank). Since the Gram matrices are exactly the PSD matrices, Eq. (9) can be simplified to:

$$\left.\begin{aligned} \min \sum_{\{i,j\} \in E} (X_{ii} + X_{jj} - 2X_{ij}) \\ \forall \{i,j\} \in E \quad X_{ii} + X_{jj} - 2X_{ij} \geq d_{ij}^2 \\ X \succeq 0. \end{aligned}\right\} \tag{11}$$

Note that objective functions for the EDGP are often (though not always [3]) a matter of preference and empirical experience on sets of instances, which makes sense since the EDGP is a pure feasibility problem (other possible objectives include adding slack variables which are then minimized). From here onwards, therefore, we shall simply discuss pure feasibility formulations expressed with equality constraints, each of which can be turned into an optimization problem at need, with equality constraints possibly changed into inequalities, and/or by additional slacks and surplus variables to be minimized.

3 Diagonally dominant programming

One serious drawback of SDP is that current solving technology is limited to instances of fairly low sizes. Ahmadi and Hall recently remarked [1] that diagonal dominance provides a useful tool for inner approximating the PSD cone. A matrix (Y_{ij}) is DD if

$$\forall i \leq n \quad Y_{ii} \geq \sum_{j \neq i} |Y_{ij}|. \tag{12}$$

It follows from Gershgorin's theorem that all DD matrices are PSD (the converse does not hold, hence the inner approximation). This means that

$$\left.\begin{aligned} \forall \{i,j\} \in E \; X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \\ Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \text{ is DD} \end{aligned}\right\} \tag{13}$$

is a DDP formulation with a feasible region which is an inner approximation of that of Eq. (8).

The crucial observation is that Eq. (12) is easy to linearize exactly, as follows:

$$\begin{aligned} \forall i \leq n \quad & \sum_{j \neq i} T_{ij} \; \leq \; Y_{ii} \\ \forall i,j \leq n \quad & -T_{ij} \leq Y_{ij} \leq T_{ij}. \end{aligned}$$

We exploit this idea to derive a new DDP formulation related to the EDGP, which is in fact an LP for the EDGP.

$$\left.\begin{aligned} \forall \{i,j\} \in E \; X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \\ \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} = Y \\ \forall i \leq n+K \qquad \sum_{\substack{j \leq n+K \\ j \neq i}} T_{ij} \leq Y_{ii} \\ -T \leq Y \leq T. \end{aligned}\right\} \tag{14}$$

Note that, previous to Eq. (14), the only existing LP formulation for the EDGP was the relaxation of Eq. (4) in which every monomial $m(x)$ of the quartic polynomial in the objective is linearized to a variable μ subject to linear convex and concave relaxations of the nonconvex constraint $\mu = m(x)$. It is known [12] that, for large enough variable bounds, this relaxation is much weaker than the obvious lower bound 0. We hope that the new formulation Eq. (14) will improve the situation.

3.1 DDP from the dual

Since Eq. (14) is an inner approximation of Eq. (8), there might conceivably be cases where the feasible region of Eq. (14) is empty while the feasible region of Eq. (8) is non-empty (quite independently of whether the original EDGP

instance has a solution or not). For such cases, Ahmadi and Hall recall that the dual of any SDP is another SDP (moreover, strong duality holds). So it suffices to derive a DDP from the dual of the SDP relaxation Eq. (8) in order to obtain a new, valid LP relaxation of the EDGP.

3.2 Iterative improvement of the DDP formulation

Ahmadi and Hall also provide an iterative method to improve the DDP inner approximation for general SDPs, which we adapt here to Eq. (14). For any symmetric $n \times n$ matrix U, we have $U^\top U \succeq 0$ since any Gram matrix is PSD. By the same reason, $U^\top X U \succeq 0$ for any $X \succeq 0$. This implies that

$$\mathcal{D}(U) = \{U^\top A U \mid A \text{ is DD}\} \tag{15}$$

is a subset of the PSD cone. We can therefore replace the constraint "Y is DD" by $Y \in \mathcal{D}(U)$ in Eq. (13). Note that this means the LP formulation is now parametrized on U, which offers the opportunity to choose U so as to improve the approximation. More precisely, we define a sequence of DDP formulations:

$$\left.\begin{aligned} \forall \{i,j\} \in E \; X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \\ Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \in \mathcal{D}(U^h), \end{aligned}\right\} \tag{16}$$

for each $h \in \mathbb{N}$, with

$$\begin{aligned} U^0 &= I \\ U^h &= \mathsf{factor}(\bar{Y}^{h-1}), \end{aligned}$$

where $\mathsf{factor}(\cdot)$ indicates a factor of the argument matrix (Ahmadi and Hall suggest using Choleski factors for efficiency), and $\bar{Y}^h$ is the solution of Eq. (16) for a given h.

The iterative method ensures that, for each h, the feasible region of Eq. (16) contains the feasible region for $h-1$. This is easily seen to be the case since, if U^h is a factor of $\bar{Y}^{h-1}$, we trivially have $(U^h)^\top I U^h = (U^h)^\top U^h = \bar{Y}^{h-1}$, and since I is trivially DD, $\bar{Y}^{h-1} \in \mathcal{D}(U^h)$. Moreover, $\bar{Y}^{h-1}$ is feasible in Eq. (16), which proves the claim.

The transformation of the constraint $Y \in \mathcal{D}(U)$ into a set of linear constraints is also straightforward. $Y \in \mathcal{D}(U)$ is equivalent to "$Y = U^\top Z U$ and Z is DD", i.e.

$$\begin{aligned} \forall i \leq n+K \quad \sum_{\substack{j \leq n+K \\ j \neq i}} T_{ij} &\leq Z_{ii} \\ -T \leq Z &\leq T \\ U^\top Z U &= Y, \end{aligned}$$

as observed above.

4 A new heuristic for the DGP

In this section we use some of the techniques discussed above in order to derive a new heuristic algorithm which will hopefully be able to solve large-scale EDGP instances.

1. Solve a DDP approximation to an SDP relaxation of the DGP (see previous sections) to yield $\bar{X}$. If $\mathsf{rank}(\bar{X}) \leq K$, factor $\bar{X} = \bar{x}\bar{x}^\top$ and return $\bar{x}$.
2. We now have $\bar{X}$ with $\mathsf{rank}(\bar{X}) > K$. We run Barvinok's randomized rank reduction algorithm [4]:
   (a) sample $y \in \mathbb{R}^{nK}$ from a multivariate normal standard distribution $\mathcal{N}^{nK}(0,1)$
   (b) let $T = \mathsf{factor}(\bar{X})$
   (c) let $x' = Ty$.
   Barvinok proves that there is concentration of measure for this type of randomized rank reduction, so that, if κ is the least number such that $m = |E| \leq n^\kappa$, there is n_0 large enough such that, if $n \geq n_0$, we have:
   $$\mathsf{Prob}\left(\forall \{i,j\} \in E \quad \mathsf{dist}(x', \mathcal{X}_{ij}) \leq c(\kappa)\sqrt{\|\bar{X}\|_2 \ln n}\right) \geq p, \qquad (17)$$
   where $\mathsf{dist}(x, \mathcal{X}_{ij})$ is the Euclidean distance from x to the manifold $\mathcal{X}_{ij}$, $\|X\|_2$ is the largest eigenvalue of $\bar{X}$, $c(\kappa)$ is a constant depending only on κ, and p is given in [4] as $p = 0.9$.
3. Call any local NLP solver with x' as a starting point, and hope to return a rank K solution $x^* \in \mathbb{R}^{nK}$ which is feasible in Eq. (3).

Note that we can actually solve an SDP relaxation of the DGP, in Step 1, rather than a DDP approximation thereof. This variant of the heuristic will be used to obtain a computational comparison in Sect. 5.

We remark that we are actually mis-using Barvinok's rank reduction algorithm, which was originally developed only for $K = 1$. Concentration of measure phenomena, however, are based on average behaviour being what one would expect; the most important part of the work is always to prove that large distortions from the mean are controllably improbable. We therefore believe we are justified in our mis-appropriation, at the risk of the probability being somewhat lower than advertised; but since we have no good estimations for c, this vagueness is not overly detrimental. Essentially, most concentration of measure results are often used qualitatively in algorithmic design, as a statement that, for large enough sizes, the expected behaviour is going to happen with ever higher probability.

On the other hand, mis-using a theoretical result is not to be taken lightly, even if justified by common sense. This is why we also obtained some additional computational experiments (not reported here) with SDPs and DDPs derived from writing realizations as vectors in $\mathbb{R}^{nK}$ rather than $n \times K$ matrices, i.e. precisely the setting of Barvinok's theorem. We found that these results yielded similar outcomes to our heuristic, but in much slower times, due to the much larger size $O(nK \times nK)$ of the involved matrices.

5 Preliminary computational assessment

We implemented the proposed heuristic in Python 2.7 and tested it on a Darwin Kernel 15.3 (MacOSX "El Capitan") running on an Intel i7 dual-core (virtual quad-core) CPU at 3.1GHz with 16GB RAM.

These are very preliminary experiments, and should be taken as a token of validation of our ideas, not as sound empirical evidence that our idea is computationally the best for the task. As concerns the task, we aim at finding solutions for Eq. (1). Although our stated motivation is to be able to solve Eq. (2), we would like our heuristic to be able to handle both equalities and inequalities, and, for this work, all we had time for was the former.

We tested two variants of our heuristic for comparison: the original one, with Step 1 solving a DDP using the iterative method, and the variant where Step 1 solves an SDP.

Our heuristic is configured as follows.

1. We solved DDP formulations with CPLEX 12.6 [10] (default configuration), which automatically exploits all the cores.
2. We implemented Barvinok's rank approximation heuristic in Python, which only runs on a single core.
3. After testing IPOpt as a local NLP solver, we switched to the implementation of L-BFGS given in the Python module `scipy.optimize` (default configuration), which seems to be faster than IPOpt on Eq. (4), an unconstrained problem.
4. The DDP sequences in the iterative method count at most four DDP approximating LPs each — the iterative procedure is interrupted as soon as a feasible SDP point is found by DDP.

For each instance and solution method we record the (scaled) largest distance error (LDE) of the solution x, defined as

$$\mathsf{lde}(x) = \max_{\{i,j\}\in E}(|\,\|x_i - x_j\|_2 - d_{ij}\,|/d_{ij}),$$

the (scaled) mean distance error (MDE)

$$\mathsf{mde}(x) = \frac{1}{|E|}\sum_{\{i,j\}\in E}(|\,\|x_i - x_j\|_2 - d_{ij}\,|/d_{ij}),$$

and the CPU time. All CPU times have been computed in Python using the `time` module. They indicate the CPU time used by the Python process as well as its spawned sub-processes (including CPLEX and the local NLP solver) to reach termination.

We tested some randomly generated instances as well as some protein instances taken from the Protein Data Bank (PDB). In the latter, only edges smaller than 5Å were kept, which is realistic w.r.t. Nuclear Magnetic Resonance (NMR) experiments.

Our first test (see Table 1) aims at solving DGPs for $K = 2$ on three groups of instances.

- Small toy instances, infeasible for $K = 2$.
- A set of instances named `euclid-`n_p, generated randomly as follows:
  1. place n points in a square, uniformly at random;
  2. generate the cycle $1, \ldots, n$ to ensure biconnectedness;
  3. for each other vertex pair i, j, decide whether $\{i, j\} \in E$ with probability p;
  4. record the Euclidean distance d_{ij} between pairs of points in E;

  obviously, all such instances are feasible.
- Two protein instances `1b03` and `1crn`, obviously infeasible for $K = 2$.

The test emphasizes the fact that small SDPs can be solved faster than a sequence of up to four DDP approximations, but that the DDP formulations cope better with increasing sizes, which is what we expected.

Instance			*LDE*		*MDE*		*CPU*	
Name	$\|V\|$	$\|E\|$	SDP	DDP	SDP	DDP	SDP	DDP
`test1`	4	6	0.06	0.06	0.03	0.03	0.14	**0.09**
`test2`	4	6	0.43	0.43	0.08	0.08	0.15	**0.10**
`test3`	4	6	0.05	0.05	0.02	0.02	0.12	**0.11**
`random-8_0.5`	8	19	**0.78**	0.99	0.16	**0.10**	0.51	**0.40**
`cl3`	10	23	2.97	2.97	0.52	0.52	0.25	**0.24**
`dmdgp-3_10`	10	24	0.90	**0.56**	**0.13**	0.14	**0.25**	0.28
`dmdgp-3_20`	20	54	0.92	0.92	0.14	0.14	1.97	**1.31**
`testrandom`	100	1008	**0.93**	0.97	0.20	0.20	**64.76**	84.55
`euclid-10_0.5`	10	26	0^*	0.78	0^*	0.13	0.4	0.4
`euclid-20_0.5`	20	111	0^*	0^*	0^*	0^*	**1.32**	3.02
`euclid-30_0.5`	30	240	0^*	0^*	0^*	0^*	**3.19**	4.11
`euclid-40_0.5`	40	429	0^*	0^*	0^*	0^*	**6.5**	9.0
`euclid-50_0.2`	50	290	0^*	0^*	0^*	0^*	6.66	**10.69**
`euclid-50_0.3`	50	412	0^*	0^*	0^*	0^*	**9.08**	13.41
`euclid-50_0.4`	50	535	0^*	0^*	0^*	0^*	**9.04**	13.98
`euclid-50_0.5`	50	642	0^*	0^*	0^*	0^*	**12.13**	15.16
`euclid-50_0.6`	50	772	0^*	0^*	0^*	0^*	**14.29**	15.88
`euclid-60_0.2`	60	407	0^*	0^*	0^*	0^*	**13.4**	18.84
`euclid-60_0.5`	60	938	0^*	0^*	0^*	0^*	**20.78**	23.16
`euclid-60_0.6`	60	1119	0^*	0^*	0^*	0^*	**23.6**	27.9
`euclid-70_0.5`	70	1212	0^*	0^*	0^*	0^*	**32.89**	134.98
`euclid-80_0.5`	80	1639	0^*	0^*	0^*	0^*	51.74	**46.21**
`euclid-90_0.5`	90	1959	0^*	0^*	0^*	0^*	85.53	**61.2**
`1b03`	89	456	0.98	**0.84**	0.08	**0.07**	65.68	**43.89**
`1crn`	138	846	1.49	**1.15**	**0.11**	0.13	587.51	**164.57**

Table 1. Tests for $K = 2$. 0^* indicates values of $O(10^{-5})$ or less.

We are not systematically testing the usefulness of Barvinok's randomized rank reduction step. Preliminary tests indicate that, only for small instances, it

can be replaced by the output x of Principal Component Analysis (PCA): factor the SDP solution $\bar{X}$ into $x = P^\top \sqrt{\Delta_K^+}$, where P is the eigenvector matrix of $\bar{X}$, and Δ_K^+ is the diagonal matrix with the K largest positive eigenvalues and zeros elsewhere. We also tried to replace sampling of y from a uniform distribution in $[-1, 1]$ instead of a multivariate normal standard one, and obtained fairly good results. But as far as we know there are no concentration of measure results based on the uniform distribution, so this may only be an effect of the concentration of measure phenomenon "kicking in" at much larger values of n than those we tested. We want to stress that these are preliminary impressions, supported only by our experience and some limited computational evidence.

Our second test (Table 2) is more realistic, and finds realizations of (feasible) protein instances in $K = 3$.

Instance			*LDE*		*MDE*		*CPU*	
Name	$\|V\|$	$\|E\|$	SDP	DDP	SDP	DDP	SDP	DDP
`C0700.odd.G`	36	308	0^*	0^*	0^*	0^*	34.96	**22.83**
`C0700.odd.H`	36	308	$\mathbf{0}^*$	6.02e-3	$\mathbf{0}^*$	2.56e-4	**24.62**	36.45
`C0150alter.1`	37	335	1.78e-4	**1.12e-4**	0^*	0^*	**26.26**	41.59
`C0080create.1`	60	681	$\mathbf{0}^*$	1.61e-4	0^*	0^*	161.64	**123.05**
`C0080create.2`	60	681	1.4e-4	$\mathbf{0}^*$	0^*	0^*	**107.11**	129.06
`1b03`	89	456	**0.28**	0.30	**0.01**	0.02	131.98	**118.82**
`1crn`	138	846	**0.38**	0.82	**0.01**	0.02	936.39	**438.27**

Table 2. Tests on proteins for $K = 3$. 0^* indicates values of $O(10^{-5})$ or less.

6 Conclusion

We propose a new heuristic algorithm for the DGP, based on diagonally-dominant programming, a randomized rank reduction algorithm, and a local NLP solver. Although our computational test set-up is prelimimnary, we believe our results are promising, and give an indication that the computational bottleneck of SDP can be overcome by diagonal dominance and LP.

References

1. Ahmadi, A., Hall, G.: Sum of squares basis pursuit with linear and second order cone programming. Tech. Rep. 1510.01597v1, arXiv (2015)
2. Alfakih, A., Khandani, A., Wolkowicz, H.: Solving Euclidean distance matrix completion problems via semidefinite programming. Computational Optimization and Applications 12, 13–30 (1999)
3. Barvinok, A.: Problems of distance geometry and convex properties of quadratic maps. Discrete and Computational Geometry 13, 189–202 (1995)

4. Barvinok, A.: Measure concentration in optimization. Mathematical Programming 79, 33–53 (1997)
5. Cucuringu, M., Singer, A., Cowburn, D.: Eigenvector synchronization, graph ridigity and the molecule problem. Information and Inference: a journal of the IMA 1, 21–67 (2012)
6. Dokmanić, I., Parhizkar, R., Ranieri, J., Vetterli, M.: Euclidean distance matrices: Essential theory, algorithms and applications. IEEE Signal Processing Magazine 1053-5888, 12–30 (Nov 2015)
7. Eren, T., Goldenberg, D., Whiteley, W., Yang, Y., Morse, A., Anderson, B., Belhumeur, P.: Rigidity, computation, and randomization in network localization. IEEE Infocom Proceedings pp. 2673–2684 (2004)
8. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42(6), 1115–1145 (1995)
9. Hendrickson, B.: The molecule problem: exploiting structure in global optimization. SIAM Journal on Optimization 5, 835–857 (1995)
10. IBM: ILOG CPLEX 12.6 User's Manual. IBM (2014)
11. Krislock, N., Wolkowicz, H.: Explicit sensor network localization using semidefinite representations and facial reductions. SIAM Journal on Optimization 20, 2679–2708 (2010)
12. Lavor, C., Liberti, L., Maculan, N.: Computational experience with the molecular distance geometry problem. In: Pintér, J. (ed.) Global Optimization: Scientific and Engineering Case Studies, pp. 213–225. Springer, Berlin (2006)
13. Lavor, C., Liberti, L., Maculan, N., Mucherino, A.: The discretizable molecular distance geometry problem. Computational Optimization and Applications 52, 115–146 (2012)
14. Liberti, L., Lavor, C., Maculan, N.: A branch-and-prune algorithm for the molecular distance geometry problem. International Transactions in Operational Research 15, 1–17 (2008)
15. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. SIAM Review 56(1), 3–69 (2014)
16. Liberti, L., Lavor, C., Mucherino, A.: The discretizable molecular distance geometry problem seems easier on proteins. In: Mucherino, A., Lavor, C., Liberti, L., Maculan, N. (eds.) Distance Geometry: Theory, Methods, and Applications. Springer, New York (2013)
17. Liberti, L., Lavor, C., Mucherino, A., Maculan, N.: Molecular distance geometry methods: from continuous to discrete. International Transactions in Operational Research 18, 33–51 (2010)
18. Majumdar, A., Ahmadi, A., Tedrake, R.: Control and verification of high-dimensional systems with dsos and sdsos programming. In: Conference on Decision and Control. vol. 53, pp. 394–401. IEEE, Los Angeles (2014)
19. Man-Cho So, A., Ye, Y.: Theory of semidefinite programming for sensor network localization. Mathematical Programming B 109, 367–384 (2007)
20. Saxe, J.: Embeddability of weighted graphs in k-space is strongly **NP**-hard. Proceedings of 17th Allerton Conference in Communications, Control and Computing pp. 480–489 (1979)
21. Yajima, Y.: Positive semidefinite relaxations for distance geometry problems. Japan Journal of Industrial and Applied Mathematics 19, 87–112 (2002)