Computational experience on Distance Geometry Problems 2.0

Claudia D'Ambrosio[1], Vu Khac Ky[1], Carlile Lavor[2], Leo Liberti[3,1]

[1] *LIX, École Polytechnique, F-91128 Palaiseau, France*
Email:{dambrosio,liberti}@lix.polytechnique.fr

[2] *IMECC, University of Campinas, Brazil*
Email:clavor@ime.unicamp.br

[3] *IBM "T.J. Watson" Research Center, Yorktown Heights, 10598 NY, USA*
Email:leoliberti@us.ibm.com

March 19, 2014

Abstract

We propose a set of formulations and reformulations of the Distance Geometry Problem, which we evaluate with both local and global off-the-shelf solvers. The local solvers are cast in a global optimization metaheuristic (Variable Neighbourhood Search) since the problem is nonconvex and non-global optima are usually of limited practical interest.

1 Introduction

Roughly every decade, it is useful to assess how generic off-the-shelf solvers perform on mathematical programming formulations of any sufficiently important problem. We commented on computational experiments of the Distance Geometry Problem (DGP) in [7]: although the paper appeared in 2006, the experiments were conducted between 2004 and 2005, so it is high time to re-evaluate the current state of the art.

The DGP requires to "draw" a given weighted graph in $\mathbb{R}^K$ in such a way that the Euclidean distances of the segments between pairs of vertices match the given edge weights. More formally, given a simple undirected graph $G = (V, E)$, an integer $K > 0$, and an edge weight function $d : E \to \mathbb{R}_+$, the DGP asks to establish or deny the existence of a vertex realization function $x : V \to \mathbb{R}^K$ such that:

$$\forall \{u, v\} \in E \qquad \|x_u - x_v\|_2 = d_{uv}. \tag{1}$$

Notationwise, we let $n = |V|$ and $m = |E|$. More information can be found in [10].

In the following, we briefly summarize the results of [7] in Sect. 2, then proceed to list the DGP formulations we shall consider (Sect. 3), the evaluation framework (Sect. 4), and the test set (in Sect. 5). The computational results will be presented at the conference.

2 The computational set-up in [7]

Our testbed for [7] was simple (*way* too simple, in fact): three cubic grid instances taken from [12], and six "protein-like" instances generated randomly according to [6]. We solved these instances using three global optimization solvers: a deterministic, ε-approximate spatial Branch-and-Bound (sBB) algorithm [8], a stochastic Multi-Start (MS) algorithm based on Sobol' sequences [5], and a Variable Neighbourhood Search (VNS) solver for nonconvex Nonlinear Programs (NLP) [9].

The solvers were launched in their default configurations to solve the following unconstrained NLP formulation of the DGP:

$$\min_{x\in\mathbb{R}^{Kn}} \sum_{\{u,v\}\in E} (\|x_u - x_v\|_2^2 - d_{uv}^2)^2 \tag{2}$$

on each instance. All cubic grid instances were solved at global optimality by all solvers, as well as four out of six protein-like instances. The remaining two instances were solved by the stochastic solvers (MS and VNS), whereas the sBB was terminated because of the CPU time threshold (1hr of user time). Only one instance failed to be solved to global optimality (i.e. within $\varepsilon = 10^{-3}$), but came nonetheless pretty close. The best overall solver was the VNS.

3 The DGP formulation zoo

First off, all formulations we consider are box-constrained (this makes life simpler for certain solvers) to $x \in [-M, M]^{Kn}$ where $M = \sum_{\{u,v\}\in E} d_{uv}$: we do not write these bounds explicitly below. Every formulation comes with variants; a variant which holds for every formulation is the following: replace $\|x_u - x_v\|_2^2$ by $\|x_u - x_v\|_2$ and d_{uv}^2 by d_{uv}. In such variants, because of floating point issues, $\sqrt{\alpha}$ is implemented as $\sqrt{\alpha+\delta}$, where δ is $O(10^{-10})$. Notationwise, $\mathbf{M} = [-M, M]^m$.

3.1 Exact formulations

1. Eq. (2). Variant: replace $\sum$ with max.
2. This formulation minimizes slacks and tries to satisfy Eq. (1):

$$\left.\begin{array}{rrcl} \min\limits_{x,s\in\mathbf{M}} & \sum\limits_{\{u,v\}\in E} s_{uv}^2 & & \\ \forall\{u,v\}\in E & \|x_u - x_v\|_2^2 & = & d_{uv}^2 + s_{uv}. \end{array}\right\} \tag{3}$$

   Variants: (i) replace s_{uv}^2 with $s_{uv}^+ + s_{uv}^-$ and s_{uv} with $s_{uv}^+ - s_{uv}^-$, where $s^+, s^- \geq 0$; (ii) replace $\sum$ with max.
3. This formulation exploits the convexity and concavity of the equations in Eq. (1) separately:

$$\left.\begin{array}{rrcl} \max\limits_{x} & \sum\limits_{\{u,v\}\in E} \|x_i - x_v\|_2^2 & & \\ \forall\{u,v\}\in E & \|x_u - x_v\|_2^2 & \leq & d_{uv}^2. \end{array}\right\} \tag{4}$$

4. This is a Nonlinear Complementarity Problem (NCP) formulation:

$$\left.\begin{array}{rrcl} \max\limits_{x,y\in\mathbf{M},z\in[0,1]^m} & \sum\limits_{\{u,v\}\in E} z_{uv} & & \\ \forall\{u,v\}\in E & \|x_u - x_v\|_2^2 & = & y_{uv} \\ \forall\{u,v\}\in E & (y_{uv} - d_{uv}^2)z_{uv} & = & 0. \end{array}\right\} \tag{5}$$

5. This exploits $\|x_u - x_v\|_2^2 = (x_u - x_v)(x_u - x_v)$:

$$\left.\begin{array}{rrcl} \min\limits_{x,\sigma\in\mathbf{M}^K,\tau\in\mathbf{M}^K} & \sum\limits_{\{u,v\}\in E}\sum\limits_{k\leq K} (\sigma_{uvk} - \tau_{uvk})^2 & & \\ \forall\{u,v\}\in E, k\leq K & x_{uk} - x_{vk} & = & \sigma_{uv} \\ \forall\{u,v\}\in E & \sum\limits_{k\leq K} \sigma_{uvk}\tau_{uvk} & = & d_{uv}^2. \end{array}\right\} \tag{6}$$

6. This is a nonsmooth version of Eq. (2):

$$\min_{x} \sum_{\{u,v\}\in E} |\|x_u - x_v\|_2^2 - d_{uv}^2|. \tag{7}$$

   Variant: replace $\sum$ with max.

3.2 Pointwise exact reformulations

These are formulations which are only exact for a specific set of values assigned to certain parameters; they can be used in a stochastic search setting (such as MS or VNS) where the global search occurs over the parameter values. The advantage is that they describe convex problems, so they are solved efficiently.

1. We use formulation 4 and rewrite the norm terms as per Item 5 in Sect. 3.2. This yields:
$$\left.\begin{array}{rl} \max\limits_{x} & = \sum\limits_{\{u,v\}\in E} \sum\limits_{k\leq K} \theta_{uvk}(x_{uk} - x_{vk}) \\ \forall \{u,v\} \in E & \|x_u - x_v\|^2 \leq d_{uv}^2 \end{array}\right\} \tag{8}$$
(it can be shown that there exist values of θ for which (8) is an exact formulation for the DGP).
2. Every formulation in Sect. 3.1 which involves the term $\|x_u - x_v\|_2^2$ gives rise to a pointwise exact convex reformulation, apart from Eq. (5) which gives rise to a Linear Complementarity Problem (LCP).
3. Eq. (7) can itself be interpreted as a pointwise exact convex reformulation if τ are taken as parameters rather than decision variables. It can be further reformulated as a pointwise exact *linear* reformulation by replacing the objective function by $\min\limits_{x,\sigma} \sum\limits_{\substack{\{u,v\}\in E \\ k\leq K}} |\sigma_{uvk} - \tau_{uvk}|$.

4 The computational evaluation framework

There is an obvious evaluation framework which consists of gathering computational measures about quality and efficiency of each solver on each formulation for each instance, and compare them on various indices: one may thus answer empirical questions such as, "what is the best solver+formulation combination for a given instance?", or "I need to find the conformation of a set of proteins given some distance data: what solver should I buy and what formulation should I use?" Of course one may also fail to answer such questions, whenever there is no clear winner.

A less trivial evaluation framework is given by the Multiplicative Weights Algorithm (MWA) [1]: each of the N solver+formulation combinations (indexed by $i = (s, f)$) is assigned a weight, which is initially set to 1. We decide an order $<$ for the instance set, and then solve each instance t in the set using every solver+formulation i in the order $<$. At the t-th iteration, we record a non-negative cost μ_{it} of the pair (i, t), which is a convex combination of the solution error and the CPU time (both scaled to the respective maxima). These costs are used to update the weights $\omega_{i,t+1} = \omega_{it}(1 - \frac{1}{2}\mu_{it})$. After all instances have been looked at, the result is a multivariate distribution $p = (\omega_{it}/\Phi)_{i,t}$ where $\Phi = \sum_{i,t} \omega_{it}$. The marginal distributions give an idea of the relative success and failures of the different methods and instances. This whole computation can be repeated for different orders $<$, chosen e.g. to always cluster instances of the same type. An interesting feature of the MWA is that it provides a relative bound on its total error:
$$\sum_t \sum_i \mu_{it} p_{it} \leq 2 \ln N + \frac{3}{2} \min_i \sum_t \mu_{it}, \tag{9}$$
which is a direct consequence of [1, Thm. 2.1] when the costs are nonnegative. In other words, the total weighted error made by all solvers+formulations over all instances is bounded above by a linear function of the best combination.

For a candidate solution $x' \in \mathbb{R}^{Kn}$, the average and maximum solution error definitions are:
$$\begin{aligned} \eta_{\text{avg}}(x') &= \frac{1}{m} \sum_{\{u,v\}\in E} |\|x'_u - x'_v\|_2 - d_{uv}| \\ \eta_{\max}(x') &= \max_{\{u,v\}\in E} |\|x'_u - x'_v\|_2 - d_{uv}|. \end{aligned}$$

5 The test set

We test the whole formulation zoo (with variants) with the following solvers: SNOPT [4] (local), IPOPT [2] (local), FILTER [3] (local), COUENNE (global). Local solvers and pointwise exact reformulations are tested in a Variable Neighbourhood Search framework as in [9] (i.e. with hyper-rectangular neighbourhoods centered around the current iterate). All solvers will be given the same maximum CPU time.

We shall consider DGP instances for $K = 1$ (application to clock synchronization [13]), $K = 2$ (application to sensor networks [11]), $K = 3$ (application to protein conformation from NMR data [10]). The MWA evaluation framework will be run on every order on $\{1, 2, 3\}$.

References

[1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8:121–164, 2012.

[2] COIN-OR. *Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT*, 2006.

[3] R. Fletcher and S. Leyffer. User manual for FILTER. Technical report, University of Dundee, UK, March 1999.

[4] P.E. Gill. *User's guide for SNOPT version 7.2*. Systems Optimization Laboratory, Stanford University, California, 2006.

[5] S. Kucherenko and Yu. Sytsko. Application of deterministic low-discrepancy sequences in global optimization. *Computational Optimization and Applications*, 30(3):297–318, 2004.

[6] C. Lavor. On generating instances for the molecular distance geometry problem. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, pages 405–414. Springer, Berlin, 2006.

[7] C. Lavor, L. Liberti, and N. Maculan. Computational experience with the molecular distance geometry problem. In J. Pintér, editor, *Global Optimization: Scientific and Engineering Case Studies*, pages 213–225. Springer, Berlin, 2006.

[8] L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK, March 2004.

[9] L. Liberti and M. Dražic. Variable neighbourhood search for the global optimization of constrained NLPs. In *Proceedings of GO Workshop, Almeria, Spain*, 2005.

[10] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, 2014.

[11] A. Man-Cho So and Y. Ye. Theory of semidefinite programming for sensor network localization. *Mathematical Programming B*, 109:367–384, 2007.

[12] J. Moré and Z. Wu. Global continuation for distance geometry problems. *SIAM Journal of Optimization*, 7(3):814–846, 1997.

[13] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30:20–36, 2011.