in J. Skåntorp and J. Kronqvist (eds.), Proceedings of STOGO, pp. 89-93, KTH Royal Institute of Technology, 2025



# A blueprint for computing with distance geometry

Leo Liberti[1], Maël Kupperschmitt[1], Ha Duy Nguyen[1]

[1] *LIX CNRS, École Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France,* {leo.liberti,mael.kupperschmitt–le-flao,ha-duy.nguyen}@polytechnique.edu

**Abstract** The equivalence between **NP**-hard problems given by polynomial reductions can be exploited to solve a problem in terms of another problem. Usually "other problems" are those in which humans model naturally, such as SAT. We propose instead to use the fundamental problem of distance geometry, i.e. drawing a graph in a $K$-dimensional Euclidean space by points and segments as long as the edge weights. The geometrical nature of this problem affords novel solution opportunities in continuous space (local search, relaxations) when solving combinatorial problems.

**Keywords:** Distance Geometry Problem, Complexity, Computation

## 1. Introduction

The DISTANCE GEOMETRY PROBLEM (DGP) is as follows: given a positive integer $K$ and a simple edge-weighted undirected graph $G = (V, E, d)$, decide if there exists a *realization* $x : V \to \mathbb{R}^K$ such that

$$\forall \{u, v\} \in E \quad \|x_u - x_v\|_2^2 = d_{uv}^2, \tag{1}$$

where $d_{uv}$ is the edge weight. If $K$ is fixed we denote the problem as DGP$_K$.

We exploit the equivalence properties of the complexity class **NP** (decision problems for which every YES instance can be certified in polynomial time) in order to solve hard problems for **NP** in terms of the DGP. We call this short paper a "blueprint" in the sense that many ideas herein are sketched and computationally untested.

# 2. NP-hardness primer

A problem $P$ is *hard* for the class **NP** if every other problem in **NP** can be *polynomially reduced* to $P$. A polynomial reduction from a problem $Q \in$ **NP** to $P$ is a polytime algorithm that transforms every YES instance of $Q$ in a YES instance of $P$ and every NO instance of $Q$ in a NO instance of $P$. Then $P$ can be no easier than $Q$ (up to a polynomial factor) since otherwise one could solve $P$ instead of $Q$ and obtain the same answer, which would make $Q$ easier than itself. From this it follows that every problem in **NP** is at most as hard as any **NP**-hard problem and that all **NP**-hard problems have the same hardness.

Polynomial reduction algorithms from $Q$ to $P$ transform problem instances of $Q$ into decision-invariant instances of $P$, a process often called *modelling*. The prototypical **NP**-hard problem is SAT[1]. In [2], the trace of the polytime certification algorithm for YES instances of **NP** was modelled by an infinite family of SAT instances that are satisfiable iff the certification algorithm succeeds. By [2], modelling SAT in terms of new problems $P$ is sufficient to prove the **NP**-hardness of $P$.

## 2.1 Categorizations of NP-hard problems

**NP**-hard problems that also belong to **NP** are called **NP**-complete. This provides a first categorization.

A second one is given by *weakly* and *strongly* **NP**-hard problems. If an **NP**-hard problem $Q$ has a partly numerical input (e.g. integers), and the polytime reduction to $P$ represents some of these integers in unary form (e.g. $n \in \mathbb{N}$ in $Q$ is transformed to a set of $n$ vertices in $P$), then $P$ is weakly **NP**-hard. If all integers are represented in binary form (e.g. $n$ is represented as a weight on an edge), then $P$ is strongly **NP**-hard. Although we apply this categorization to the problems, it really applies to reductions.

The third categorization is the one between *general* and *specific* problems: humans naturally use the former modelling, such as SAT, MATHEMATICAL PROGRAMMING (MP), CONSTRAINT PROGRAMMING (CP). General problems usually involve sentences with variables. Problems that are perceived to lack these properties are called specific. Variables, however, are not crucial to mathematics [7]. The description length is more important: modelling a specific problem by a general (e.g. $k$-CLIQUE in MP) leads to reasonably short descriptions of the resulting MP instance, whereas encoding a BI-

[1] The SAT problem asks the question: is a given conjunction of clauses each of which is a disjunction of literals satisfiable?

NARY LINEAR PROGRAMMING (BLP)[2] instance by $\text{DGP}_1$ leads to a longer description w.r.t. the original instance size.

## 2.2 The DGP is NP-hard

In [5] we find several reductions to the DGP, both weak and hard. In particular, the $\text{DGP}_1$ is weakly **NP**-hard by reduction from PARTITION[3] and strongly **NP**-hard by reduction from 3SAT[4].

The $\text{DGP}_1$ is known to be **NP**-complete while this is not know for the $\text{DGP}_K$ for $K > 1$ [1]. In $K > 1$ dimensions an instance with integer edge weights might lead to a realization that necessarily involves real algebraic numbers, such as e.g. a triangle graph with unit weights. We do not know a polytime algorithm for checking whether Eq. (1) holds if $x_u, x_v$ have real algebraic components. The DGP is considered a specific problem.

# 3. Motivations

The reason why we believe that there could be advantages in modelling by DGP is the geometric nature of the realization $x$: (i) although we do not know how to use $x$ to certify a YES instance precisely, it can be used approximately (e.g. using floating-point operations) to verify Eq. (1) to any desired accuracy; (ii) the fact that $x$ ranges over $\mathbb{R}^K$ reduces any combinatorial problem $Q \in \mathbf{NP}$ to one in continuous space, paving the way for the use of solution algorithms involving gradients and hessians; (iii) obtaining a relaxation of a $\text{DGP}_K$ instance may be as simple as solving the instance in a higher-dimensional space.

# 4. Contributions

## 4.1 Saxe's reduction from 3SAT to $\text{DGP}_1$

Our first contribution is a proof that the strongly polynomial reduction from 3SAT to $\text{DGP}_1$ (with edge weights restricted to $\{1, \ldots 4\}$) in [5] is valid. The original construction is based on three weighted graph gadgets, the most complicated of which represents the $j$-th clause $L_{j1} \vee L_{j2} \vee L_{j3}$ of the given 3SAT. Saxe writes:

[2] A subclass of MP with linear objective and constraints and binary variables only.
[3] Given a list $(a_1, \ldots, a_n)$ is there $I \subseteq [n] = \{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?
[4] Like SAT, but every clause has exactly 3 literals.

> careful study of the graph [...] will reveal that it is impossible to embed it in the lin in such a way that [...] all three of the $L_{jk}$ are sent to $-1$ (FALSE), but if one or more of the $L_{jk}$ are to be sent to 1 (TRUE) then [...] exactly one such embedding is possible.

In fact, the clause gadget is wrong: $L_{j2}$ is placed at the wrong vertex, and one of the edges is missing its weight. The technical report [6] corrects the gadget but still lacks a proof, which is nontrivial. We were able to write a satisfactory proof based on showing that realized cycles in the graphs fail to close correctly[5] unless at least one literal $L_{jk}$ is placed at $x_{L_{jk}} = 1$.

## 4.2 Strong reduction from BLP to DGP$_1$

Our second contribution is a reduction from BLP to DGP$_1$ based on § 4.1. First, we reduce BLP to a SAT: this involves turning all negative integers to positive and replace corresponding variables to negative literals, transform conditional additions to adder circuits, and introduce an integer comparison circuit. Then the circuit sequence is turned into a SAT instance [3], which is then transformed to a 3SAT by successive "linearization" [4]. Finally, the 3SAT is reduced to DGP$_1$ by § 4.1.

## 4.3 Weak relaxed reduction from BLP to DGP$_1$

Our third contribution is a "relaxed reduction" from BLP to DGP$_1$. Each of the $m$ inequalities $A_i y \leq b_i$ of the BLP is transformed to an equation via a binary encoding of a slack variable $s_i$ as $\sum_{h<\log_2 U_i} 2^h \beta_h$ where $U_i \geq \sum_j |A_{ij}|$ and each $\beta \in \{0, 1\}$. The equations are penalized by squaring them and then summed together. Each squared variable $\nu^2$ is replaced by $\nu$ since all variables are binary. Cross products $\nu\mu$ are linearized to a single variable $\xi$. In this form, the instance is equivalent to a SUBSET-SUM[6], which can be easily reduced to PARTITION. We then apply the weak reduction to DGP$_1$ in [5][7]. This BLP$\rightarrow$DGP$_1$ reduction is relaxed because, upon solving the DGP$_1$, the vertex positions corresponding to the $\xi$ variables may take values different from the corresponding product $\nu\mu$. We address this issue via a solution algorithm for the DGP$_1$, called Branch-and-Prune in 1D (BP$_1$), which can correct such errors at runtime[8].

[5] I.e. the first vertex position $x_\alpha$ on the cycle path is different from the last vertex position $x_\omega$.
[6] Given a list $(a_1, \ldots, a_n)$ of integers and a value $b$, is there an $I \subseteq [n]$ such that $\sum_{i \in I} a_i = b$?
[7] In the original paper as well as in [6] the proof is only sketched: we wrote a full proof.
[8] BP$_1$ places the next vertex left or right of the previous one (vertex orders allowing this always exist when $K = 1$ and the graph is connected). When two out of the three variables $\nu, \mu, \xi$ are fixed to certain values in a branch, the third value is fixed: BP$_1$ prunes the other branch.

### 4.4 Geometric relaxations

We apply the strong reduction to an infeasible BLP instance to obtain an infeasible $\text{DGP}_1$. We consider the error formulation[9] $\min\{\sum_{\{u,v\}\in E}(s^+_{uv} + s^-_{uv}) \mid \|x_u - x_v\|_2^2 = d^2_{uv}\}$. We show that globally solving this instance with $K = 2$ in increasingly thin slabs around the first coordinate leads to increasingly tightened relaxations tending towards the exact optimal value of the $\ell_1$ error. We define some convex relaxations of MINERR-DGP.

## 5. Conclusion

Much work remains to be done: (1) streamlining our reductions; (2) improving our $\text{BP}_1$ implementation; (3) tight and efficient geometric relaxations; (4) mapping relaxation bound values from the DGP back to the BLP. Lastly, we are also looking at possible reductions based on computability rather than complexity.

## References

[1] N. Beeker, S. Gaubert, C. Glusa, and L. Liberti. Is the distance geometry problem in NP? In A. Mucherino, C. Lavor, L. Liberti, and N. Maculan, editors, *Distance Geometry: Theory, Methods, and Applications*, pages 85–94. Springer, New York, 2013.

[2] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Symposium on the Theory of Computing*, STOC, pages 151–158, New York, 1971. ACM.

[3] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–25, 2006.

[4] R. Karp. Reducibility among combinatorial problems. In R. Miller and W. Thatcher, editors, *Complexity of Computer Computations*, volume 5 of *IBM Research Symposia*, pages 85–104, New York, 1972. Plenum.

[5] J. Saxe. Embeddability of weighted graphs in $k$-space is strongly NP-hard. *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.

[6] J. Saxe. Two papers on graph embedding problems. Technical Report CMU-CS-80-102, Dept. of Computing, Carnegie-Mellon University, 1980.

[7] A. Tarski and S. Givant. *A formalization of set theory without variables*. Number 41 in American Mathematical Society Colloquium Publications. AMS, Providence, RI, 1987.

[9] Called MINERR-DGP.