Seppo Pulkkinen

Incremental Low-Rank SDP Approach to Finding Graph Embeddings

Turku Centre *for* Computer Science

TUCS Technical Report
No 1069, March 2013

Incremental Low-Rank SDP Approach to Finding Graph Embeddings

Seppo Pulkkinen
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
seppo.pulkkinen@utu.fi

TUCS Technical Report
No 1069, March 2013

Abstract

Finding a low-dimensional embedding of a graph of n nodes in $\mathbb{R}^d$ is an essential task in many applications. For instance, maximum variance unfolding (MVU) is a well-known dimensionality reduction method that involves solving this problem. The standard approach is to formulate the embedding problem as a semidefinite program (SDP). However, the SDP approach does not scale well to large graphs. In this paper, we exploit the fact that many graphs have an intrinsically low dimension, and thus the optimal matrix resulting from the solution of the SDP has a low rank. This observation leads to a quadratic reformulation of the SDP that has far fewer variables, but on the other hand, is a difficult convex maximization problem. We propose an approach for obtaining a solution to the SDP by solving a sequence of smaller quadratic problems with increasing dimension. By utilizing an interior-point algorithm for solving the quadratic problems, we demonstrate by numerical experiments on MVU problems that our approach scales well to very large graphs.

Keywords: graph embedding; dimensionality reduction; maximum variance unfolding; semidefinite programming; concave quadratic programming; interior-point methods

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

Let $G = (V, E)$ be an undirected *graph* with *nodes* $V = \{1, 2, \ldots, n\}$ and *edges* $E \subset V \times V$. The problem of *isometrically embedding* the graph G into some d-dimensional space $\mathbb{R}^d$ is to assign each node i a point $\boldsymbol{y}_i \in \mathbb{R}^d$ such that (euclidean) distances between adjacent nodes are preserved. That is, given the lengths D_{ij} for each edge $\{i, j\} \in E$, the points $\boldsymbol{y}_i$ are required to satisfy the condition $\|\boldsymbol{y}_i - \boldsymbol{y}_j\| = D_{ij}$. In particular, we are interested in finding an embedding that reveals the intrinsic, possibly low dimension of the graph. As discussed, for instance in [12], [28] and [24], such an embedding can be obtained by maximizing pairwise distances between the points $\boldsymbol{y}_i$ under the above neighbour distance constraints.

In this paper, we consider relaxed graph embeddings that maximize pairwise distances of the output points $\boldsymbol{y}_i$ under two constraints. First, rather than exactly preserving the distances, which would in general be impossible, we only impose upper bounds on the distances between adjacent nodes. Second, we require that the barycenter of the points $\boldsymbol{y}_i$ is at the origin. Maximization of pairwise point distances under these two constraints leads to the *quadratically constrained quadratic program* (QCQP)

$$\max_{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n \in \mathbb{R}^d} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \boldsymbol{y}_i = \boldsymbol{0}, \tag{1b}$$

$$\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \leq D_{ij}^2, \quad \{i, j\} \in E \tag{1c}$$

with some embedding dimension d. We will show that under mild assumptions, this relaxed graph embedding problem has a well-defined solution.

For solving graph embedding problems of the form (1), the standard approach is to consider a semidefinite reformulation as, for instance, in [28]. By introducing the matrix $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^T$ with $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1, & \boldsymbol{y}_2, & \ldots, & \boldsymbol{y}_n \end{bmatrix}^T \in \mathbb{R}^{n \times d}$, problem (1) can be reformulated as the *semidefinite program* (SDP)

$$\max_{\boldsymbol{K} \in \mathcal{S}^n} \quad \operatorname{tr}(\boldsymbol{K}) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{K} \succeq 0, \tag{2b}$$

$$K_{ii} - 2K_{ij} + K_{jj} \leq D_{ij}^2, \quad \{i, j\} \in E, \tag{2c}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} K_{ij} = 0, \tag{2d}$$

where $\mathcal{S}^n$ denotes the cone of symmetric $n \times n$ matrices.

Problem (1) and its semidefinite formulation (2) appear in pattern recognition and machine learning (see e.g. [22] and [28]). Namely, based on the above

ideas, Weinberger and Saul [28] developed the well-known *maximum variance unfolding* (MVU) method for dimensionality reduction. Given a set of input points $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, a relaxed form of the MVU method described in [28] solves a special case of problem (1). In the MVU problem, the edge set E is constructed by adding an edge between each input point $\boldsymbol{x}_i$ and its k-*nearest* neighbours and also between the k-nearest neighbours of each input point. An example of such a graph is given in Figure 1. The distances D_{ij} in the MVU problem are distances between adjacent input points (i.e. $D_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ for $\{i, j\} \in E$).

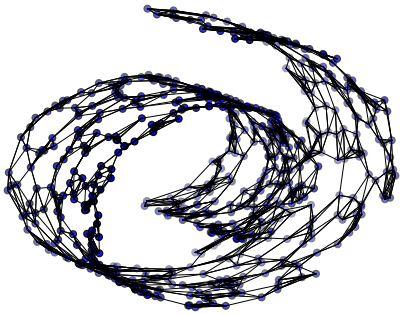


Figure 1: k-nearest neighbour graph of an example dataset.

Reformulation of the difficult convex maximization problem (1) as the SDP (2) has the advantage that it is a concave maximization problem over a convex set, and thus any local maximum is a global one. Moreover, the SDP formulation eliminates the need of knowing the embedding dimension d a priori. Once the SDP has been solved, an embedding can be obtained from the eigenvectors of the matrix $\boldsymbol{K}$ corresponding to few largest eigenvalues, as shown for instance in [28]. The standard interior-point SDP solvers are applicable to this problem (see e.g. [5], [6], [23], [25] and [33]). Unfortunately, they scale poorly to large problems since the SDP (2) has $\mathcal{O}(n^2)$ variables. Moreover, the interior-point SDP solvers involve factorization and storage of a dense Schur complement matrix of size $m \times m$, where m is the number of constraints in the problem. Since the number of edges can be significantly larger than the number of nodes in the graph (see e.g. Table 1 in Section 4), this step incurs a major computational bottleneck particularly when solving the MVU problem.

In this paper we develop a scalable approach for solving the graph embedding SDP (2). Based on the theory of semidefinite programs and their low-rank formulations developed in [9], [10], [11] and [15], we show that problem (1) is in fact a low-rank formulation of problem (2). Motivated by this fact, we develop an incremental low-rank method for solving the SDP (2). The idea of the method is to solve a sequence of small quadratic problems (1) with increasing dimension d until a solution to the SDP is obtained. Furthermore, we show that such a solution is globally optimal for (1) with dimension d that gives an upper bound for the rank of the optimal solution of (2). Due to zero duality gap, which we will establish for the SDP (2) and its dual under mild assumptions, we also obtain a solution to the

dual problem.

Theoretically, the incremental low-rank approach has computational advantages over the interior-point SDP solvers and the low-rank SDP solver by Burer and Monteiro [9,10] and Burer and Choi [8]. The former methods attempt to solve the SDP directly, and the latter solves quadratic problems of larger dimension and reduces the dimension as necessary. Hence, the advantage is particularly large when solving the SDP (2) for a graph having a low-dimensional embedding. As a solver for the quadratic problems, we use `IPOPT`, an interior point filter-based line search algorithm by Wächter and Biegler [30,31]. As we will show, another major factor contributing to the good performance of our approach is the ability of `IPOPT` to exploit second-order information and sparsity in the highly structured graph embedding problem.

Utilizing `IPOPT` in our low-rank framework, we conduct a series of numerical experiments on MVU problems of the form (2) constructed from synthetically generated datasets. By these experiments, we demonstrate that by formulating the MVU problem (and also more general graph embedding problems) as a sequence of smaller quadratic problems, the solution can be obtained far more efficiently than by using traditional SDP solvers. We also show that the `IPOPT` algorithm incorporated into our low-rank framework is more efficient than the first-order L-BFGS algorithm used in the low-rank method of Burer et al. Previously, the method by Burer et al. has been applied to the MVU problem in [16] and [27].

The remainder of this paper is organized as follows. In Section 2, we provide the theoretical results for the incremental approach to solving the SDP (2) via a sequence of low-dimensional problems (1). A practical implementation based on the results of Section 2 and the `IPOPT` algorithm is described in Section 3. Section 4 is devoted to numerical tests demonstrating the computational efficiency of our approach. Finally, in Section 5 we conclude the paper and point out some directions of future research.

2 Low-Rank Approach and Optimality Conditions

In this section, we recall the main results concerning the relation between optimal solutions of semidefinite programs and their low-rank reformulations. We will show that problem (1) is a low-rank reformulation of problem (2), which allows us to apply the standard results. These results will be utilized in Section 3, where we develop an algorithm that is guaranteed to give a solution of the SDP (2) by solving a sequence of smaller problems (1).

2.1 Optimality Conditions

In what follows, we recall the main results by Burer and Monteiro [9,10], Grippo et al. [11] and Journée at al. [15]. These results give necessary and sufficient

conditions that a (local) solution to the QCQP

$$\begin{aligned} \min_{\boldsymbol{Y}\in\mathbb{R}^{n\times d}} \quad & \boldsymbol{C}\bullet(\boldsymbol{Y}\boldsymbol{Y}^T) \\ \text{s.t.} \quad & \boldsymbol{A}_i\bullet(\boldsymbol{Y}\boldsymbol{Y}^T)=b_i, \quad i=1,2,\ldots,m \end{aligned} \tag{3}$$

arising from the change of variables $\boldsymbol{K}=\boldsymbol{Y}\boldsymbol{Y}^T$ with $\boldsymbol{Y}\in\mathbb{R}^{n\times d}$ such that $d\leq n$ is also a solution to the standard form SDP

$$\begin{aligned} \min_{\boldsymbol{K}\in\mathcal{S}^n} \quad & \boldsymbol{C}\bullet\boldsymbol{K} \\ \text{s.t.} \quad & \boldsymbol{K}\succeq 0, \\ & \boldsymbol{A}_i\bullet\boldsymbol{K}=b_i, \quad i=1,2,\ldots,m. \end{aligned} \tag{4}$$

Here the operator $\bullet$ between two $n\times n$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as

$$\boldsymbol{A}\bullet\boldsymbol{B}=\text{tr}(\boldsymbol{A}^T\boldsymbol{B})=\sum_{i,j=1}^{n}a_{ij}b_{ij}.$$

A key assumption made in [11] is that problem (4) and its dual have nonempty solution sets and the gap between the primal and dual solutions of the SDP (4) is zero. The dual of problem (4) is

$$\begin{aligned} \max_{\boldsymbol{\lambda}\in\mathbb{R}^m} \quad & \boldsymbol{b}^T\boldsymbol{\lambda} \\ \text{s.t.} \quad & \boldsymbol{C}-\sum_{i=1}^{m}\lambda_i\boldsymbol{A}_i\succeq 0. \end{aligned} \tag{5}$$

Assumption 2.1. *Problem* (4) *and its dual* (5) *have nonempty solution sets. In addition, if* $\boldsymbol{K}^*\in\mathbb{R}^{n\times n}$ *is a solution of* (4) *and* $\boldsymbol{\lambda}^*\in\mathbb{R}^m$ *is a solution of the dual problem* (5)*, then* $\boldsymbol{C}\bullet\boldsymbol{K}^*=\boldsymbol{b}^T\boldsymbol{\lambda}^*$.

In the following, we shall interchangeably use the matrix $\boldsymbol{Y}^*$ and the vector $\boldsymbol{y}^*$ that are related according to

$$\boldsymbol{Y}^*=\begin{bmatrix} y_1^* & y_{n+1}^* & \cdots & y_{nd-n+1}^* \\ y_2^* & y_{n+2}^* & \cdots & y_{nd-n+2}^* \\ \vdots & \vdots & \ddots & \vdots \\ y_n^* & y_{2n}^* & \cdots & y_{nd}^* \end{bmatrix}\in\mathbb{R}^{n\times d}. \tag{6}$$

For a vector $\boldsymbol{y}^*\in\mathbb{R}^{nd}$ obtained by stacking the columns of the matrix $\boldsymbol{Y}^*$ defined by equation (6), we shall use the notation $\text{vec}(\boldsymbol{Y}^*)$, and for a matrix $\boldsymbol{Y}^*$ obtained from a vector $\boldsymbol{y}^*\in\mathbb{R}^{nd}$ according to equation (6) we shall use the notation $\text{mat}(\boldsymbol{y}^*)$.

In order to state the optimality conditions for the QCQP (3) in the standard vector notation, we consider the problem

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^{nd}} \quad & \boldsymbol{y}^T(\boldsymbol{I}_d \otimes \boldsymbol{C})\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{y}^T(\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y} = b_i, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{7}$$

Here $\boldsymbol{I}_d$ denotes the $d \times d$ identity matrix and the symbol $\otimes$ denotes the *Kronecker product*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{nm}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{mp\times nq}$$

between two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{p\times q}$. Problems (3) and (7) are equivalent by the identity

$$\boldsymbol{A} \bullet (\boldsymbol{B}\boldsymbol{B}^T) = \boldsymbol{b}^T(\boldsymbol{I}_m \otimes \boldsymbol{A})\boldsymbol{b} \tag{8}$$

for any matrices $\boldsymbol{A} \in \mathcal{S}^n$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and the vector $\boldsymbol{b} = \text{vec}(\boldsymbol{B})$.

Under Assumption 2.1, Grippo et al. [11] give the following sufficient condition for a solution of (7) to be a solution of the SDP (4) and also a global solution to (7). For the KKT optimality conditions, we refer to [4].

Theorem 2.1. *Under Assumption 2.1, if $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a first-order KKT point of (7) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ and the condition*

$$\boldsymbol{C} + \sum_{i=1}^{m} \lambda_i^* \boldsymbol{A}_i \succeq 0 \tag{9}$$

holds, then the matrix $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$, where $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$, is a solution to (4) and $\boldsymbol{y}^*$ is a global solution to (7).

Conversely, the following necessary condition is established in [11].

Theorem 2.2. *Under Assumption 2.1, if $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a global optimum of (7) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ and the matrix $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$ with $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$ is a solution to (4), then condition (9) holds.*

Journée et al. [15] derive an alternative sufficient condition for a solution of problem (7) to be a solution to the SDP (4). By generalizing earlier results derived in [9] and [11], they show that if the matrix $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$, where $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a solution to problem (7), is rank deficient (i.e. $\text{rank}(\boldsymbol{Y}^*) < d$), then the matrix $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$ is a solution to (4). The result holds under the assumption that the constraint gradients at the optimal solution are linearly independent.

Assumption 2.2. *If $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a solution to (7), then the constraint gradients $2(\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y}^*$, $i = 1, 2, \ldots, m$, are linearly independent.*

Theorem 2.3. *Let $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ be a second-order KKT point of (7) such that the matrix $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*) \in \mathbb{R}^{n\times d}$ satisfies the condition $\text{rank}(\boldsymbol{Y}^*) < d$. Then the matrix $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$ is a solution to (4).*

Finally, Burer and Monteiro [10] give a lower bound for d ensuring that any local solution $\boldsymbol{Y}^*$ of (3) yields a solution $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$ of (4). They show that this holds when

$$d \geq \bar{d} = \max\{d \in \mathbb{N} \mid \frac{d(d+1)}{2} \leq m+1\}. \tag{10}$$

As we will see in Section 4, this bound is rather conservative for the MVU problem (and likely for most graph embedding problems), and optimal solutions of problem (2) can in practice be obtained by solving problems (1) with a much smaller dimension d. Nevertheless, this result gives a theoretical guarantee of finite termination for the algorithm described in Section 3.

2.2 Matrix Formulation of the Embedding Problem

Problems (1) and (2) are not in the standard forms (3) and (4), but they can be stated in this form. To this end, let $G = (V, E)$ be a graph with n nodes and n_E edges and let

$$E = ((i_1, j_1), (i_2, j_2), \ldots, (i_{n_E}, j_{n_E})) \tag{11}$$

denote an ordered sequence of the edges. Further, let us define the matrices

$$\boldsymbol{C} = -\boldsymbol{I}_n \quad \text{and} \quad \boldsymbol{A}_k = \boldsymbol{a}_k\boldsymbol{a}_k^T, \quad k = 1, 2, \ldots, n_E, \tag{12}$$

where the vectors $\boldsymbol{a}_k \in \mathbb{R}^n$ are defined as

$$a_{k,l} = \begin{cases} 1, & \text{if } l = i_k, \\ -1, & \text{if } l = j_k, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

In addition, let us denote by $\mathbf{1}_n$ a vector of ones having length n and define $b_k = D^2_{i_k,j_k}$ for $k = 1, 2, \ldots, n_E$. Then by a straightforward calculation we obtain the following results. The proof of Proposition 2.1 is given in Appendix B, and Proposition 2.2 follows directly from the above definitions.

Proposition 2.1. *A set of vectors $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathbb{R}^d$ is feasible for problem (1) if and only if the matrix $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1, & \boldsymbol{y}_2, & \ldots, & \boldsymbol{y}_n \end{bmatrix}^T \in \mathbb{R}^{n\times d}$ is feasible for the problem*

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{n\times d}} \quad \boldsymbol{C} \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) \tag{14a}$$

$$\text{s.t.} \quad \boldsymbol{A}_i \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) \leq b_i, \quad i = 1, 2, \ldots, n_E, \tag{14b}$$

$$(\mathbf{1}_n\mathbf{1}_n^T) \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) = 0. \tag{14c}$$

Furthermore, for any feasible $\boldsymbol{Y}$ *we have*

$$\boldsymbol{C} \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) = -\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$$

and the constraint functions of problems (14) *and* (1) *have the same values.*

Proposition 2.2. *A matrix* $\boldsymbol{K} \in \mathcal{S}^n$ *is feasible for problem* (2) *if and only if it is feasible for the problem*

$$\begin{aligned} \min_{\boldsymbol{K}\in\mathcal{S}^n} \quad & \boldsymbol{C} \bullet \boldsymbol{K} \\ \text{s.t.} \quad & \boldsymbol{K} \succeq 0, \\ & \boldsymbol{A}_i \bullet \boldsymbol{K} \leq b_i, \quad i = 1, 2, \ldots, n_E, \\ & (\mathbf{1}_n\mathbf{1}_n^T) \bullet \boldsymbol{K} = 0. \end{aligned} \tag{15}$$

Furthermore, for any feasible $\boldsymbol{K}$ *we have* $\boldsymbol{C} \bullet \boldsymbol{K} = -\text{tr}(\boldsymbol{K})$ *and the constraint functions of problems* (15) *and* (2) *have the same values.*

Propositions 2.1 and 2.2 essentially state that the QCQP (1) is a low-rank formulation of the SDP (2). Furthermore, in Appendix C we show that problems (14) and (15) can be equivalently stated in the equality-constrained forms (3) and (4), respectively, by introducing *slack variables*. Finally, in Appendix A we show that under mild assumptions problem (2) and its dual have nonempty solution sets, and the optimal objective function values of the primal and dual problems coincide. By this fundamental property, Assumption 2.1 is satisfied and the results of this section are applicable to the SDP (2) and the QCQP (1).

3 The Algorithm and Implementation

Based on the theoretical discussion of Section 2, we now describe the algorithmic framework for obtaining a solution to the graph embedding SDP (2) by means of the smaller QCQP (1). The idea is to successively solve problem (1) with increasing dimension d until a solution of problem (2) is attained. We also give a short introduction to the `IPOPT` algorithm by Wächter and Biegler [30, 31] and discuss how to employ it in the proposed framework.

In order to solve problem (1) via the low-rank formulation (14) by a nonlinear optimization algorithm, we need to state problem (14) in the standard vector notation. By identity (8), we can equivalently state this problem as

$$\min_{\boldsymbol{y}\in\mathbb{R}^{nd}} \quad \boldsymbol{y}^T(\boldsymbol{I}_d \otimes \boldsymbol{C})\boldsymbol{y} \tag{16a}$$

$$\text{s.t.} \quad \boldsymbol{y}^T(\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y} \leq b_i, \quad i = 1, 2, \ldots, n_E, \tag{16b}$$

$$\boldsymbol{y}^T[\boldsymbol{I}_d \otimes (\mathbf{1}_n\mathbf{1}_n^T)]\boldsymbol{y} = 0. \tag{16c}$$

Unfortunately, the Hessian of the Lagrangian of problem (16) is dense due to the dense matrix $\mathbf{1}_n\mathbf{1}_n^T$ appearing in the constraint (16c). Thus, it is essential to note that constraint (16c) can be formulated as a (small) set of linear constraints. This property can be observed by using the identity

$$\boldsymbol{I}_d \otimes (\mathbf{1}_n\mathbf{1}_n^T) = (\boldsymbol{I}_d \otimes \mathbf{1}_n)(\boldsymbol{I}_d \otimes \mathbf{1}_n)^T,$$

which yields

$$\boldsymbol{y}^T[\boldsymbol{I}_d \otimes (\mathbf{1}_n\mathbf{1}_n^T)]\boldsymbol{y} = \|(\boldsymbol{I}_d \otimes \mathbf{1}_n)^T\boldsymbol{y}\|^2 = \sum_{i=1}^{d}(\boldsymbol{y}^T\boldsymbol{c}_i^d)^2 = 0 \tag{17}$$

with

$$c_{i,j}^d = \begin{cases} 1, & \text{if } j = (i-1)n+1, (i-1)n+2, \ldots, (i-1)n+n, \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

for $i = 1, 2, \ldots, d$. From equation (17), we then observe that condition (16c) is equivalent to the condition that $\boldsymbol{y}^T\boldsymbol{c}_i^d = 0$ for all $i = 1, 2, \ldots, d$.

By the above remarks, Proposition 2.1 and the equivalence between problems (14) and (16), we can now state problem (1) in the standard NLP form as

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^{nd}} \quad & f(\boldsymbol{y}) \\ \text{s.t.} \quad & g_i^d(\boldsymbol{y}) \le b_i, \qquad i = 1, 2, \ldots, n_E, \\ & h_i^d(\boldsymbol{y}) = 0, \qquad i = 1, 2, \ldots, d, \end{aligned} \tag{NLP$_d$}$$

where

$$\begin{aligned} f(\boldsymbol{y}) &= -\|\boldsymbol{y}\|^2, \\ g_i^d(\boldsymbol{y}) &= \boldsymbol{y}^T(\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y}, \qquad i = 1, 2, \ldots, n_E, \\ h_i^d(\boldsymbol{y}) &= \boldsymbol{y}^T\boldsymbol{c}_i^d, \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{19}$$

This problem is equivalent to problem (1) up to the scaling of the objective function (cf. Proposition 2.1). This is a concave minimization problem under convex constraints, and thus the feasible set is convex. Furthermore, in Appendix A we state the assumptions under which the feasible set is bounded.

3.1 Incremental Low-Rank Algorithm

In this subsection, we describe an algorithm that successively solves a sequence of low-rank quadratic problems (NLP$_d$) starting with some small dimension $d = d_0 \ge 1$ and increasing d if the solution is not optimal to the graph embedding SDP (2). Derivation of the algorithm involves some technical considerations that we defer to Appendix B.

The following theorem, whose proof is given in Appendix B, gives a computationally convenient form of the stopping criterion (9) adapted to the graph embedding SDP (2) and its quadratic low-rank formulation (NLP$_d$). For the sequel, we introduce the function $\kappa : \mathcal{S}^n \to \mathbb{R}$ to denote the second smallest eigenvalue (among the n, not necessarily distinct eigenvalues) of a symmetric $n \times n$ matrix.

Theorem 3.1. *Let $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ be a solution to (NLP$_d$) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$ corresponding to the constraints $g_i^d(\boldsymbol{y}^*) \leq b_i$. The matrix $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$ with $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$ is a solution to (2) and $\boldsymbol{y}^*$ is a global solution of (NLP$_d$) if and only if the condition*

$$\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) \geq 1 \tag{20}$$

is satisfied, where

$$\boldsymbol{L}(\boldsymbol{\lambda}) = \sum_{i=1}^{n_E} \lambda_i \boldsymbol{A}_i$$

and the matrices $\boldsymbol{A}_i$ are defined according to equations (12) and (13).

Adapting condition (9) to the graph embedding problem is computationally more convenient than adapting the rank deficiency condition provided by Theorem 2.3. Unlike the rank deficiency condition, condition (9) does not require solving problem (NLP$_d$) with d greater than the rank of the optimal solution of the SDP (2). Furthermore, when the matrix $\boldsymbol{L}(\boldsymbol{\lambda}^*)$ appearing in condition (20) is sparse (which is the case for MVU graphs), an efficient Lanczos-type algorithm (see e.g. [18]) can be used to compute its second smallest eigenvalue.

If by solving problem (NLP$_d$) we obtain a vector $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ for which the matrix $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$ with $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$ is not a solution of the SDP (2) by condition (20), then we increase the dimension d by one and solve problem (NLP$_{d+1}$). A natural choice is to augment the vector $\boldsymbol{y}^*$ with zeros and choose $\tilde{\boldsymbol{y}}^* = [\boldsymbol{y}^{*^T}\ \boldsymbol{0}_n^T]^T$ as the starting point for the solution of problem (NLP$_{d+1}$). The following theorem, whose proof is given in Appendix B, states that $\tilde{\boldsymbol{y}}^*$ is a saddle point of problem (NLP$_{d+1}$) and gives a descent direction from $\tilde{\boldsymbol{y}}^*$. This direction is also feasible (i.e. it is orthogonal to the constraint gradients at $\tilde{\boldsymbol{y}}^*$).

Theorem 3.2. *Let $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ be a first-order KKT point of problem (NLP$_d$) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$, $\boldsymbol{\lambda}^* \geq \boldsymbol{0}$, and $\boldsymbol{\mu}^* \in \mathbb{R}^d$ corresponding to the constraints $g_i^d(\boldsymbol{y}^*) \leq b_i$ and $h_i^d(\boldsymbol{y}) = 0$, respectively. If condition (20) is not satisfied, then the vectors*

$$\tilde{\boldsymbol{y}}^* = \begin{bmatrix} \boldsymbol{y}^* \\ \boldsymbol{0}_n \end{bmatrix}, \quad \boldsymbol{\lambda}^* \quad \textit{and} \quad \tilde{\boldsymbol{\mu}}^* = \begin{bmatrix} \boldsymbol{\mu}^* \\ 0 \end{bmatrix}$$

satisfy the first-order KKT conditions of problem (NLP$_{d+1}$). Furthermore, the eigenspace corresponding to the eigenvalue $\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*))$ contains an eigenvector

$\boldsymbol{v}^*$ such that $\mathbf{1}_n^T \boldsymbol{v}^* = 0$. With such a vector $\boldsymbol{v}^*$, the Hessian of the Lagrangian $\mathcal{L}(\boldsymbol{y}; \boldsymbol{\lambda}; \boldsymbol{\mu})$ of problem (NLP$_{d+1}$) satisfies the condition $\boldsymbol{d}^T \nabla_{\boldsymbol{y}}^2 \mathcal{L}(\tilde{\boldsymbol{y}}^*; \boldsymbol{\lambda}^*; \tilde{\boldsymbol{\mu}}^*) \boldsymbol{d} < 0$ with

$$\boldsymbol{d} = \begin{bmatrix} \mathbf{0}_{nd} \\ \boldsymbol{v}^* \end{bmatrix}.$$

In addition, the direction $\boldsymbol{d}$ satisfies the conditions

$$\nabla g_i^{d+1}(\tilde{\boldsymbol{y}}^*)^T \boldsymbol{d} = 0, \quad i = 1, 2, \ldots, n_E,$$
$$\nabla h_i^{d+1}(\tilde{\boldsymbol{y}}^*)^T \boldsymbol{d} = 0, \quad i = 1, 2, \ldots, d+1.$$

Hence, from a saddle point $\tilde{\boldsymbol{y}}^*$ a solution to problem (NLP$_{d+1}$) can be obtained by perturbing the solution along the descent direction $\boldsymbol{d}$ and using a descent method from the starting point

$$\boldsymbol{y}_0 = \tilde{\boldsymbol{y}} + \varepsilon \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|}, \quad \text{where } \tilde{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y}^* \\ \mathbf{0}_n \end{bmatrix} \quad \text{and} \quad \boldsymbol{d} = \begin{bmatrix} \mathbf{0}_{nd} \\ \boldsymbol{v}^* \end{bmatrix}$$

with some small $\varepsilon > 0$.

The meta-algorithm for solving the SDP (2) by the incremental low-rank approach is listed as Algorithm 1. The algorithm shares some similarities with the generic incremental low-rank algorithm of [15], but we have derived it under less restrictive assumptions and specifically for the graph embedding problem. In particular, our algorithm only requires Assumption 2.1 that is satisfied by the graph embedding problem (2) and its dual under mild assumptions. Finally, we note that by the bound (10), Algorithm 1 is guaranteed to give a solution to the SDP (2) after a finite number of steps.

Algorithm 1: Incremental Low-Rank Graph Embedding.

input : graph matrices $\{\boldsymbol{A}_i\}_{i=1}^{n_E} \subset \mathbb{R}^{n \times n}$
squared edge lengths $\boldsymbol{b} \in \mathbb{R}^{n_E}$ such that $\boldsymbol{b} > \mathbf{0}$
initial solution dimension $d_0 > 0$
initial solution $\boldsymbol{Y}_0 \in \mathbb{R}^{n \times d_0}$
perturbation parameter $\varepsilon > 0$

output: embedding $\boldsymbol{Y}^* \in \mathbb{R}^{n \times d}$ with some $d \geq d_0$

$d \leftarrow d_0$

$\boldsymbol{y}_0 \leftarrow \text{vec}(\boldsymbol{Y}_0)$; $\quad \boldsymbol{\lambda}_0 \leftarrow \mathbf{0}$

Starting from $\boldsymbol{y}_0 \in \mathbb{R}^{nd}$ and $\boldsymbol{\lambda}_0 \in \mathbb{R}^{n_E}$, obtain $\boldsymbol{y}^*$, a second-order KKT point of (NLP$_d$) with Lagrange multipliers $\boldsymbol{\lambda}^*$.

$\boldsymbol{L}^* \leftarrow \sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i$

Compute eigenvalue $\kappa(\boldsymbol{L}^*)$ and eigenvector $\boldsymbol{v}^*$ according to Theorem 3.2.

if $\kappa(\boldsymbol{L}^*) \geq 1$ **then**

  Terminate with $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$.

else

  $\boldsymbol{d} \leftarrow [\mathbf{0}_{nd}^T \quad \boldsymbol{v}^{*T}]^T$

  $\boldsymbol{y}_0 \leftarrow [\boldsymbol{y}^{*T} \quad \mathbf{0}_n^T]^T + \varepsilon \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|}$; $\quad \boldsymbol{\lambda}_0 \leftarrow \boldsymbol{\lambda}^*$

  $d \leftarrow d + 1$

3.2 The IPOPT Interior-Point Algorithm

IPOPT implements a primal-dual interior point method that uses a filter-based line-search strategy [30,31]. For an overview of nonlinear interior-point and filter methods, we refer to [19]. IPOPT is particularly designed for solving large-scale nonlinear optimization problems with second-order derivatives and with a known sparsity structure. Internally, IPOPT transforms a given NLP into the form

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^n} \quad & f(\boldsymbol{y}) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{y}) = \boldsymbol{0}, \\ & l_i \le y_i \le u_i, \quad i \in I \end{aligned} \tag{21}$$

with constraints $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$, some index set $I \subset \{1, 2, \ldots, m\}$ and lower and upper bounds l_i and u_i, respectively. For instance, problem (NLP$_d$) with inequality constraints can be reformulated in this way by introducing n_E additional slack variables.

IPOPT enforces bound constraints via a logarithmic *barrier function*. In order to solve the NLP (21), the algorithm (approximately) solves a sequence of *barrier problems*

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^n} \quad & \varphi_\mu(\boldsymbol{y}) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{y}) = \boldsymbol{0} \end{aligned} \tag{22}$$

with

$$\varphi_\mu(\boldsymbol{y}) = f(\boldsymbol{y}) - \mu \sum_{i\in I} \ln(y_i)$$

and a sequence of parameters $\mu > 0$ converging to zero.

For solving the barrier problem (22), IPOPT uses a linearization of its KKT conditions. With the current primal and dual iterates $\boldsymbol{y}_k$ and $\boldsymbol{\lambda}_k$ for the solution of (21), a linear approximation of the KKT conditions yields the system

$$\begin{bmatrix} \boldsymbol{W}_k + \boldsymbol{\Sigma}_k + \delta_{\boldsymbol{w}}\boldsymbol{I} & \boldsymbol{J}_k \\ \boldsymbol{J}_k^T & -\delta_c \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_k^{\boldsymbol{y}} \\ \boldsymbol{d}_k^{\boldsymbol{\lambda}} \end{bmatrix} = - \begin{bmatrix} \nabla\varphi_\mu(\boldsymbol{y}_k) + \boldsymbol{J}_k\boldsymbol{\lambda}_k \\ \boldsymbol{h}(\boldsymbol{y}_k) \end{bmatrix} \tag{23}$$

for the primal and dual search directions $\boldsymbol{d}_k^{\boldsymbol{y}}$ and $\boldsymbol{d}_k^{\boldsymbol{\lambda}}$. These search directions are used in a filter-based line search algorithm.

In the KKT system (23), the matrix $\boldsymbol{W}_k = \nabla^2_{\boldsymbol{y}}\mathcal{L}(\boldsymbol{y}_k; \boldsymbol{\lambda})$ denotes the Hessian of the Lagrangian with respect to $\boldsymbol{y}$. For problem (NLP$_d$), the Hessian of the Lagrangian

$$\mathcal{L}(\boldsymbol{y}; \boldsymbol{\lambda}) = -\|\boldsymbol{y}\|^2 + \sum_{i=1}^{n_E} \lambda_i [\boldsymbol{y}^T (\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y} - b_i] + \sum_{i=1}^{d} \lambda_{n_E+i} \boldsymbol{y}^T \boldsymbol{c}_i^d$$

is the $nd \times nd$-matrix

$$\nabla^2_{\boldsymbol{y}}\mathcal{L}(\boldsymbol{y}; \boldsymbol{\lambda}) = -2\boldsymbol{I}_{nd} + 2\sum_{i=1}^{n_E} \lambda_i (\boldsymbol{I}_d \otimes \boldsymbol{A}_i). \tag{24}$$

The matrix $\boldsymbol{\Sigma}_k$ is a diagonal matrix with nonnegative elements, and $\boldsymbol{J}_k = \boldsymbol{J}(\boldsymbol{y}_k)$ denotes the Jacobian of the constraints at $\boldsymbol{y}_k$. For the constraint functions defined according to (19), the Jacobian is the $(n_E + d) \times nd$-matrix

$$\boldsymbol{J}(\boldsymbol{y}) = 2 \begin{bmatrix} \boldsymbol{B}(\boldsymbol{y}) \\ \boldsymbol{C} \end{bmatrix}, \quad \boldsymbol{B}(\boldsymbol{y}) = \begin{bmatrix} [(\boldsymbol{I}_d \otimes \boldsymbol{A}_1)\boldsymbol{y}]^T \\ [(\boldsymbol{I}_d \otimes \boldsymbol{A}_2)\boldsymbol{y}]^T \\ \vdots \\ [(\boldsymbol{I}_d \otimes \boldsymbol{A}_{n_E})\boldsymbol{y}]^T \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} (\boldsymbol{c}_1^d)^T \\ (\boldsymbol{c}_2^d)^T \\ \vdots \\ (\boldsymbol{c}_d^d)^T \end{bmatrix}. \quad (25)$$

The parameters $\delta_{\boldsymbol{w}} \geq 0$ and $\delta_{\boldsymbol{c}} \geq 0$ control the amount of regularization that is needed to ensure nonsingularity of the matrix of the KKT system (23).

Typically, either solution of the KKT system (23) or evaluation of the objective function is the computationally most expensive step in the `IPOPT` algorithm. Fortunately, both steps can be carried out at a low computational cost when solving problem (NLP$_d$). Due to the structure of the matrices $\boldsymbol{A}_i$, we observe from equation (24) that the Hessian $\boldsymbol{W}_k$ has at most $(n + n_E)d$ distinct nonzero elements. Having at most $(n + 2n_E)d$ nonzero elements, the Jacobian $\boldsymbol{J}_k$ defined by equation (25) is also sparse when n_E, the number of edges depends linearly on n, the number of nodes in the graph. As seen from Table 1 in Section 4, in MVU problems n_E depends linearly on n. Consequently, the KKT system (23) can be efficiently solved due to sparsity of the Hessian and Jacobian matrices. As demonstrated in Section 4, this reflects to the good scalability of `IPOPT` when applied to large MVU problems. On the other hand, as evaluation of the objective function $f(\boldsymbol{y}) = -\|\boldsymbol{y}\|^2$ requires only $\mathcal{O}(nd)$ operations, its computational cost is negligible compared to the solution of the KKT system.

4 Numerical Results

In this section, we demonstrate the computational efficiency of our incremental low-rank approach for solving (relaxed) MVU problems that are special cases of the general graph embedding problem (2). We compare our method to the interior-point SDP solvers `CSDP` [6] and `SDPA` [33] as well as to `SDPLR` [9, 10] that is an earlier method based on a quadratic low-rank reformulation.

4.1 Datasets and Test Setup

For eight synthetically generated datasets with n points, we constructed a *k-neighbourhood graph* as in the MVU method (see [28]). The k-neighbourhood of a given point picked from a point set in $\mathbb{R}^d$ contains the point itself and its k nearest neighbours (measured in Euclidean distance).

Definition 4.1. *Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$. The k-neighbourhood of a*

point $\boldsymbol{x} \in \boldsymbol{X}$ *is*

$$\mathcal{N}_{\boldsymbol{x},k} = \{\{\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_{k+1}}\} \subset \boldsymbol{X} \mid i_j \neq i_l \quad \forall j \neq l, \\ \|\boldsymbol{y} - \boldsymbol{x}\| \geq \|\boldsymbol{x}_{i_j} - \boldsymbol{x}\| \quad \forall \boldsymbol{y} \in \boldsymbol{X} \setminus \mathcal{N}_{\boldsymbol{x},k}, \quad j = 1, 2, \ldots, k+1\}.$$

In the k-neighbourhood graph of a given point set in $\mathbb{R}^d$, two points are connected by an edge if the other is a k-nearest neighbour of the other one, or if they lie in the k-neighbourhood of another point.

Definition 4.2. *Let* $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$*. The* k*-neighbourhood graph of* $\boldsymbol{X}$ *is the graph* $G_{\boldsymbol{X},k} = (V, E)$ *with* $V = \{1, 2, \ldots, n\}$ *and*

$$E = \{\{i, j\} \subset V \mid \exists \boldsymbol{z} \in \boldsymbol{X} : \{\boldsymbol{x}_i, \boldsymbol{x}_j\} \in \mathcal{N}_{\boldsymbol{z},k}\}.$$

The test datasets and their low-dimensional representations obtained by solving problem (2) for their k-neighbourhood graphs are shown in Appendix D. The values of k for the test datasets and the number of edges n_E for the k-neighbourhood graphs with some values of n are listed in Table 1. All tests were run on a 3.0GHz Core 2 Duo processor running a 64-bit Linux operating system and with 6GB of system memory. For the SDP solvers that implement parallelization, we used both cores of the dual-core processor in our test system, and for `SDPLR` and `IPOPT` we used one core.

	k	n	n_E		k	n	n_E
Helix	6	800	4800	Incomplete tire	6	800	5702
		1600	9600			1600	11548
		2500	15000			2500	18457
		4000	24000			4000	29764
S-roll	6	800	5933	Spiral	15	800	14899
		1600	11839			1600	32354
		2500	18741			2500	52408
		4000	29690			4000	65605
Swiss roll	6	800	5807	Trefoil knot	9	800	6611
		1600	11653			1600	13083
		2500	18451			2500	20607
		4000	29560			4000	32695
Trefoil ribbon	9	800	8065	Twin peaks	6	800	5889
		1600	17665			1600	11688
		2500	28137			2500	18582
		4000	46001			4000	29666

Table 1: Neighbourhood sizes k and the number of edges n_E in the k-neighbourhood graphs constructed from the test datasets.

4.2 Solvers

In this subsection, we give a brief description of the solvers included in our numerical tests. We also describe our modifications to their default parameters and consider some implementation details.

4.2.1 IPOPT-MVU

In the following, we refer to as `IPOPT-MVU` Algorithm 1 with `IPOPT` version 3.10.2 as the solver for problems (NLP$_d$). In all tests, we set the initial dimension d_0 to one and chose the starting point $\boldsymbol{y}_0 \in \mathbb{R}^{nd}$ randomly such that

$$y_{0,i} \in [-a, a], \text{ where } a = \min_{i=1,2,\ldots,n_E} \frac{\sqrt{b_i}}{2}$$

in order to guarantee a feasible starting point. Due to the random choice of the starting point, all computation times for `IPOPT-MVU` reported in the following are measured as an average from ten repeated test runs. For the `IPOPT` algorithm, we set all stopping criterion thresholds to 10^{-5}.

According to our observations, the choice of the algorithm for solving the KKT system (23) largely determines the performance of the overall algorithm. For solving the KKT system, we used Harwell `MA57` [1] which in our preliminary experiments was faster than its predecessor `MA27` used by Wächter and Biegler [31] and the default solver `MUMPS` [2] used by `IPOPT`. In order to improve performance of `MA57`, we disabled automatic scaling of the linear system by setting the parameters `linear_system_scaling` and `ma57_automatic_scaling` to `none` and `no`, respectively.

Instead of using the default Fiacco-McCormick strategy for updating the barrier parameter μ, we set the parameter `mu_strategy` to `adaptive` to accelerate convergence. The adaptive update strategy implemented in `IPOPT` is described in detail in [20]. In addition, we used the warm-start strategy provided by the algorithm to initialize solution of (NLP$_{d+1}$) from a solution of (NLP$_d$) with the current Lagrange multipliers. We implemented the calculation of the second smallest eigenvalue of the matrix appearing in the stopping criterion (20) by using `ARPACK` in the shift invert mode [17].

4.2.2 Interior-point SDP Solvers

`CSDP` [6] is an interior-point method for solving general semidefinite programs of the form (4). The algorithm, whose theoretical background is explained in [13], solves the original problem via a sequence of barrier problems by taking successive predictor and corrector steps. In our tests, we used `CSDP` version 6.1.1. Since the main application of the MVU method is to produce low-dimensional representations for visualization purposes, we slightly modified the default parameters of

CSDP to improve performance at some expense of solution accuracy. Thus, we relaxed the stopping thresholds from their default values and set

$$\texttt{axtol} = 10^{-4}, \quad \texttt{atytol} = 10^{-4} \quad \text{and} \quad \texttt{objtol} = 10^{-4},$$

where the meaning of these parameters is explained in the CSDP manual. We also set the parameter fastmode to 1 in order to improve performance at the expense of some solution accuracy.

SDPA [33] is another variant of an interior-point SDP solver using a Mehrotra-type primal-dual predictor-corrector method. In our tests we used version 7.3.8. As with CSDP, we relaxed the default stopping criterion and set epsilonStar $= 10^{-4}$, where the meaning of this parameter is explained in the SDPA manual.

Theoretically, the interior-point SDP solvers have very high computational complexity and storage requirements, which is consistent with our experiments in Subsection 4.3. For instance, the computational cost of each iteration of CSDP for a problem with sparse constraint matrices with $\mathcal{O}(1)$ nonzero elements (as in the MVU problem) is $\mathcal{O}(mn^2 + m^2 + m^3 + n^3)$, where m is the number of constraints and n is the size of the matrix to be solved (see [6]). This is even without the matrix $\mathbf{1}_n\mathbf{1}_n^T$ appearing in the constraints of problem (15). Since neither of the solvers can exploit low-rank structure in the constraints, they treat this matrix as a dense one, which also contributes to their poor scalability. The high storage requirements, in turn, stem from the fact that the interior-point SDP solvers require storage of a dense $m \times m$ Schur complement matrix.

4.2.3 SDPLR

In addition to the standard SDP solvers, we compare the performance of our approach with the SDPLR algorithm by Burer and Monteiro [9] with improvements by Burer and Choi [8]. As our method, SDPLR solves a given SDP of the form (4) by solving a sequence of quadratic low-rank problems (3).

Differently to our approach that uses an interior-point method, SDPLR solves the constrained problem (3) by solving a sequence of unconstrained problems, where the objective function is the *augmented Lagrangian*

$$\mathcal{L}(\boldsymbol{Y}; \boldsymbol{\lambda}; \sigma) = \boldsymbol{C} \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) + \boldsymbol{\lambda}^T[\boldsymbol{b} - \mathcal{A}(\boldsymbol{Y}\boldsymbol{Y}^T)] + \frac{\sigma}{2}\|\boldsymbol{b} - \mathcal{A}(\boldsymbol{Y}\boldsymbol{Y}^T)\|^2 \quad (26)$$

with some *penalty parameter* $\sigma > 0$. Here the operator $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ for a matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is defined as

$$[\mathcal{A}(\boldsymbol{B})]_i = \boldsymbol{A}_i \bullet \boldsymbol{B}, \quad i = 1, 2, \ldots, m.$$

At each main iteration with fixed $\boldsymbol{\lambda}_k$ and σ_k, SDPLR minimizes the augmented Lagrangian $\mathcal{L}(\boldsymbol{Y}; \boldsymbol{\lambda}_k; \sigma_k)$ with respect to the matrix $\boldsymbol{Y}$. For solving this subproblem, the algorithm uses the L-BFGS method. The L-BFGS method is implemented so that it exploits both sparsity and low-rank structure present in the

problem to improve performance. As a result, the computational complexity of one iteration of SDPLR for solving the MVU problem is linear with respect to n (see [9] for complexity analysis of SDPLR). Furthermore, as shown in [8], the minima of the quartic function (26) along a given search line have a closed-form expression, which allows the algorithm to use an exact line search. After each minimization of the augmented Lagrangian, the Lagrange multipliers $\boldsymbol{\lambda}_k$ and the penalty parameter σ_k are then updated by a heuristic method based on an infeasibility measure. Convergence analysis for the SDPLR algorithm is given in [10].

For minimization of the augmented Lagrangian (26) with given $\boldsymbol{\lambda}$ and σ, SDPLR uses the stopping criterion

$$\frac{\|\nabla_{\boldsymbol{Y}} \mathcal{L}(\boldsymbol{Y}_k; \boldsymbol{\lambda}; \sigma)\|_F}{1 + C_{\max}} \leq \frac{\rho_c}{\sigma}$$

with some $\rho_c > 0$, where

$$C_{\max} = \max_{i,j=1,2,\ldots,n} |C_{ij}|$$

and k denotes iteration index of the L-BFGS method in the inner loop of the algorithm. For the MVU problem, we have $C_{\max} = 1$. As a stopping criterion for the main iteration, SDPLR uses the scaled infeasibility measure

$$\frac{\|\boldsymbol{b} - \mathcal{A}(\boldsymbol{Y}_k \boldsymbol{Y}_k^T)\|}{1 + |b_{\max}|} \leq \rho_f$$

with some $\rho_f > 0$, where

$$b_{\max} = \max_{i=1,2,\ldots,m} |b_i|.$$

As for the SDP solvers, we modified the default threshold for the main stopping criterion ρ_f. Furthermore, we observed that a significant performance increase can be obtained by relaxing the threshold value ρ_c for the inner loop iterations. Thus, we chose $\rho_f = 10^{-2}$ and $\rho_c = 10^5$, and for the L-BFGS Hessian approximation we used the six most recent iterations. In our tests we used SDPLR version 1.03-beta.

For choosing the dimension d of the matrix $\boldsymbol{Y}$ in the augmented Lagrangian subproblems, Burer and Monteiro propose an approach where d is initially chosen as $\bar{d}$ defined by equation (10). In [8], Burer and Choi describe a heuristic method for reducing d, which is motivated by a rank deficiency condition similar to the one provided by Theorem 2.3. However, since this heuristic for choosing d is rather conservative for our application, for each test run of SDPLR we solved problem (3) with fixed dimension $d = 4$. This choice reflects the best-case situation where an estimate of the intrinsic dimension of the input data is known a priori. It is also motivated by our observation that SDPLR performs better when d is slightly larger than the rank of the optimal solution (see Table 4 in Subsection 4.4). The solutions obtained in this way are not guaranteed to be solutions of the SDP (2), but we numerically verified that all computation times reported in the following correspond to solutions that are close to the SDP solution.

4.2.4 Other Implementation Details

It is essential to use optimized linear algebra packages to achieve the best performance with the interior-point SDP solvers. Thus, for the dense linear algebra involved in CSDP and SDPA, we used ATLAS (Automatically Tuned Linear Algebra Software) version 3.8.4 [29] and LAPACK version 3.4.1 [3]. In addition, for CSDP, SDPA and SDPLR, we converted the inequality-constrained problems (15) into equality-constrained form by introducing n_E additional slack variables. For CSDP and SDPA, the test problems were supplied in the sparse SDPA format, and for SDPLR the test problems were supplied in the SDPLR format that extends the SDPA format by adding support for low-rank constraint matrices.

4.3 Performance Comparison

The computation times used by the tested solvers on the eight datasets are listed in Table 2 with dataset sizes $n = 800$, 1600, 2500 and 4000. The times are measured in seconds, and they represent wall-clock times (excluding the time consumed for reading the problem file). From the results of Table 2 we can clearly observe that the solvers fall in three performance classes, which becomes more apparent when the dataset size n is large.

First of all, these results clearly reveal the inherent limitations of the SDP approach. Despite the highly optimized linear algebra packages and parallelization available in our test system, the interior-point SDP solvers CSDP and SDPA are clearly the slowest. Unfolding many datasets with only 2500 points took several hours with both solvers, which cannot be considered an acceptable performance for real-world applications. Moreover, these solvers have high memory requirements. As a result, we were not able to run the tests for these solvers with $n = 4000$ because our test system ran out of memory. We can also observe that SDPA is generally slightly faster than CSDP. However, a detailed inspection of the solver outputs revealed that SDPA had numerical difficulties especially for larger datasets, which led to premature termination in several cases. This problem did not appear as often with CSDP, which indicates that it is more reliable.

SDPLR performs significantly better than the interior-point SDP solvers. However, on some datasets, the computation times of SDPLR seem to grow rapidly as the dataset size n is increased. This could be explained by slow convergence due to the fact that SDPLR does not use second-order information even when it is available in the test problems. Furthermore, it does not implement as sophisticated machinery to prevent ill-effects due to numerical instability as IPOPT does. This may cause SDPLR to run into numerical difficulties especially when n is large. On the other hand, IPOPT-MVU that uses a robust second-order algorithm, is significantly faster, which can be seen with all dataset sizes n. The additional cost of solving the KKT system and computing the Hessian of the Lagrangian and the constraint Jacobian, which is low due to their sparsity, is seemingly compensated

		Algorithm			
Dataset	n	CSDP	SDPA	SDPLR	IPOPT-MVU
Helix	800	70.70	73.92	13.62	6.10
	1600	1022.38*[1]	499.43*	8.85	15.12
	2500	6003.98	3067.02*	19.31	19.93
	4000	-[2]	-	116.93	91.38
Incomplete tire	800	233.28	196.37	13.40	2.62
	1600	2086.04	1575.87	39.79	8.89
	2500	9030.10	7794.99	267.70	15.14
	4000	-	-	1284.64	18.57
S-roll	800	293.89	213.85	7.86	2.72
	1600	2175.45	1675.20	17.42	7.63
	2500	9605.56	6925.95	58.44	21.23
	4000	-	-	251.11	42.94
Spiral	800	5060.78	2217.75*	16.86	11.54
	1600	-	-	140.19	44.23
	2500	-	-	917.11	125.93
	4000	-	-	6222.59	69.88
Swiss roll	800	279.52	209.26	18.09	4.03
	1600	2262.69	1514.16*	56.83	10.36
	2500	10898.32	6053.26*	311.31	18.35
	4000	-	-	1740.16	32.98
Trefoil knot	800	726.50*	203.20*	4.62	3.61
	1600	5298.22*	1421.72*	23.29	9.31
	2500	-	5255.61*	79.89	20.75
	4000	-	-	1043.15	45.98
Trefoil ribbon	800	816.47	483.68	30.38	3.60
	1600	6866.26	5970.95*	472.91	11.12
	2500	-	-	1992.96	20.04
	4000	-	-	7231.15	44.81
Twin peaks	800	242.64	207.46	4.47	2.76
	1600	1886.06	1774.86	15.08	11.13
	2500	8783.87	6771.08	44.75	19.85
	4000	-	-	359.10	26.43

[1] "*" means that the solution was obtained with reduced accuracy due to premature termination.

[2] "-" means that the solver either ran out of memory or failed to reach the stopping criterion in five hours.

Table 2: Computation times used by the solvers for unfolding the test datasets.

by the faster convergence of IPOPT. Moreover, differently to SDPLR we observe that IPOPT-MVU has consistent performance on all datasets. Another point of interest is that the computation time used by IPOPT-MVU does not always increase with n. This is because the ranks of the optimal MVU solutions may vary depending on n (cf. Table 4 in Subsection 4.4).

In order to test the scalability of IPOPT-MVU to very large datasets, we measured its computation times on the test datasets[1] with n up to 75000. The wall-clock times used by IPOPT-MVU (in seconds) are listed in Table 3. They are computed as average values of successful runs (stopping criterion reached in five hours) from ten repeated attempts. No computation time is given for datasets where IPOPT-MVU either ran out of memory or did not succeed in any of the ten test runs. For three datasets with $n = 75000$, IPOPT-MVU was able to solve the MVU problem in less than one hour, which is a remarkable achievement.

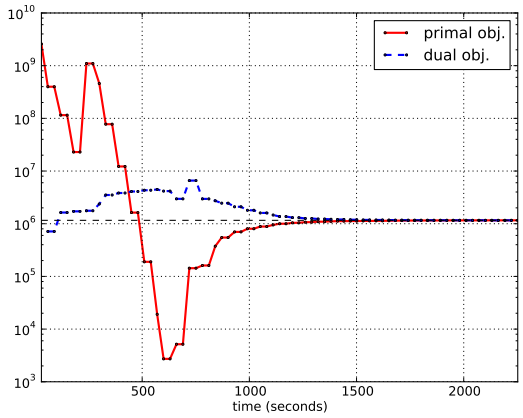
	n					
Dataset	2500	5000	12500	25000	50000	75000
Helix	19.93	147.17	763.37	1680.82	-	-
Incomplete tire	15.14	36.94	149.91	294.99	1711.94	1184.98
S-roll	21.23	66.03	352.63	404.63	3037.08	-
Spiral	125.93	105.10	910.69	-	-	-
Swiss roll	18.35	44.38	171.68	395.13	820.17	2984.63
Trefoil ribbon	20.04	52.28	238.01	589.64	1819.24	-
Twin peaks	19.85	59.69	217.86	922.71	1458.66	3246.52

Table 3: Computation times used by IPOPT-MVU for unfolding the test datasets with large number of samples n.

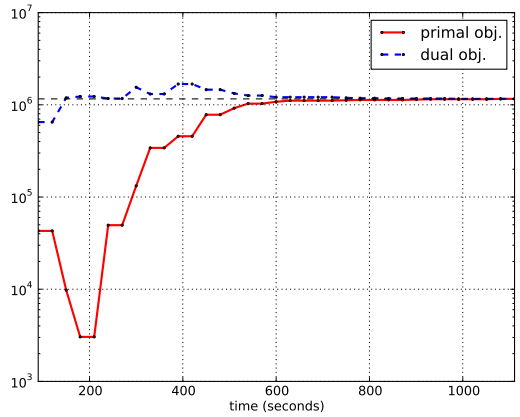
We again emphasize that IPOPT-MVU has superior performance compared to the other solvers, since unfolding datasets with over 4000 points turned out to be computationally intractable for them. By using the interior-point SDP methods, solution of large MVU problems would not be possible due to the high computational complexity and memory requirements, and by using SDPLR it would not be possible because of the apparent failure to converge (at least when an accurate solution is required).

When carrying out a numerical comparison of optimization algorithms, it is important to keep in mind that the performance of the solvers may crucially depend on the chosen stopping criterion threshold. This point is particularly relevant considering that MVU is typically used for visualization purposes where approximate solutions may be sufficient. In order to shed more light on the convergence behaviour of the tested solvers, Figure 2 shows their objective function values plotted as a function of computation (wall-clock) time on the Swiss roll

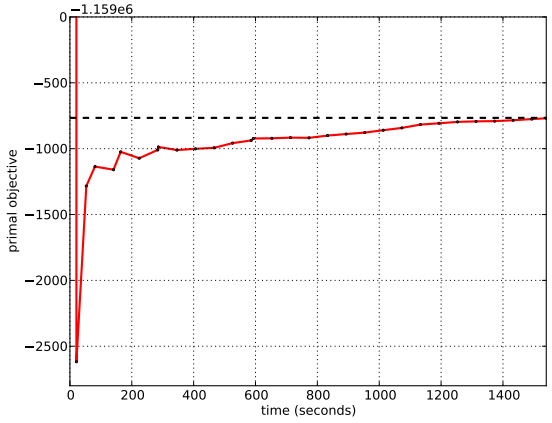
[1]The Trefoil knot dataset is not included in these tests since the corresponding MVU problem did not have a bounded solution with large n.



(a) CSDP



(b) SDPA



(c) SDPLR

Figure 2: Convergence rates of `CSDP`, `SDPA` and `SDPLR` on the Swiss roll dataset with $n = 1600$. The dashed line represents the optimal objective function value obtained by `IPOPT-MVU`.

dataset with $n = 1600$. The optimal optimal objective function value obtained with IPOPT−MVU is plotted as a dashed line. This line can be considered to represent the "exact" solution since IPOPT−MVU generally gives very accurate solutions with our stopping criterion thresholds. The convergence plots obtained for the other datasets follow similar trends, and since the observations discussed below apply to them as well, we do not show the other plots here.

From Figure 2 we can draw two conclusions. The first observation is that performance of the interior-point SDP solvers cannot be significantly improved by relaxing the stopping criterion. The primal and dual objective function values for both CSDP and SDPA stabilize only after 1200 and 600 seconds, respectively. These times are already significantly larger than the times used by SDPLR and IPOPT−MVU for the Swiss roll dataset with $n = 1600$ (cf. Table 2).

The second observation is that SDPLR converges rapidly to an approximate solution, but its convergence rate severely degrades when the solution is approached. This observation motivates our choice for a rather loose stopping criterion $\rho_f = 10^{-2}$ for SDPLR since it may be nevertheless competitive with IPOPT−MVU when only an approximate solution to the MVU problem is sought. However, choosing a stopping criterion that yields a "satisfactory" solution seems to be a matter of trial and error. On the other hand, due to the inability of SDPLR to produce accurate solutions, it cannot be used directly in our low-rank framework. This is because our approach in practice requires accurate solutions to problems (NLP$_d$) with accurate Lagrange multiplier estimates.

4.4 Ranks of MVU Solutions

Finally, we numerically verify our hypothesis that solution of the MVU problem has low rank when the input dataset is intrinsically low-dimensional. To this end, we give the ranks of the optimal solutions obtained by IPOPT−MVU for all the test datasets with four different sizes. Another point of interest is the maximum value of the dimension d used by IPOPT−MVU. Namely, the strategy for escaping a saddle point of (NLP$_d$) may involve solution of the problem with d greater than the actual rank of the optimal solution.

The ranks of the optimal solutions of problems (2) obtained with IPOPT−MVU for each dataset are listed in Table 4. For all datasets, the rank r^* of the optimal solution of the SDP (2) is close to, if not exactly, the intrinsic dimensionality of the dataset, which is two (cf. Appendix D). We also observed from our numerical experiments that the maximum dimension d used by IPOPT−MVU is in all cases the same as r^*. Consequently, a solution of the SDP (2) can be efficiently obtained by our low-rank method when the intrinsic dimension of the input data is low.

Our finding about the maximum dimension d is quite surprising considering the fact that as a concave minimization problem, for a given d, problem (NLP$_d$) is expected to have a large number of local minima that are not necessarily global. A more extensive testing with a large number of different starting points and with

$d \geq r^*$ revealed, though, that the IPOPT solver can in some cases, but rarely, converge to a non-global minimum of problem (NLP$_d$) such that the corresponding matrix $\boldsymbol{K}^*$ is not a solution to the SDP (2).

Dataset	n	r^*	Dataset	n	r^*
Helix	800	4	Incomplete tire	800	2
	1600	4		1600	3
	2500	4		2500	3
	4000	4		4000	2
S-roll	800	2	Spiral	800	3
	1600	2		1600	3
	2500	3		2500	3
	4000	3		4000	2
Swiss roll	800	3	Trefoil knot	800	2
	1600	3		1600	2
	2500	3		2500	2
	4000	3		4000	2
Trefoil ribbon	800	2	Twin peaks	800	2
	1600	2		1600	3
	2500	2		2500	3
	4000	2		4000	2

Table 4: Ranks r^* of the optimal solutions of the MVU problems for the test datasets.

	σ_1	σ_2	σ_3	σ_4
Helix	233.71	233.71	$4.82 \cdot 10^{-2}$	$4.82 \cdot 10^{-2}$
Incomplete tire	146.88	40.23	$4.74 \cdot 10^{-2}$	–
S-roll	105.49	57.85	–	–
Spiral	48.36	1.03	0.17	–
Swiss roll	1059.82	191.17	2.73	–
Trefoil knot	155.82	131.55	–	–
Trefoil ribbon	146.43	136.75	–	–
Twin peaks	34.68	26.98	$6.84 \cdot 10^{-4}$	–

Table 5: The four largest nonzero singular values of the MVU solution matrices for the test datasets with $n = 1600$.

Finally, to gain further insight on the dimensionality of the solutions of the MVU problem, we examine the eigenvalues of the solution matrices $\boldsymbol{K}^*$ of problem (2) for the test datasets. Instead of computing the eigenvalues for the full matrices, we list the singular values of the smaller matrices $\boldsymbol{Y}^*$ resulting from the solutions of the quadratic reformulations of problem (2) with IPOPT-MVU

in Table 5. These singular values are square roots of the eigenvalues of the SDP solution matrices $\boldsymbol{K}^* = \boldsymbol{Y}^*\boldsymbol{Y}^{*^T}$.

Though the MVU solutions do not exactly reveal the intrinsic dimension of all datasets (which would be the case if only the first two singular values were nonzero), there is a clearly distinguishable gap between the second and third largest singular value in all test cases. The small nonzero singular values are expected in practice due to noise or numerical inaccuracy in the input data. This is also due to the fact that distance measurements from insufficiently sampled or incomplete data do not exactly represent the (geodesic) distances on the manifold containing the input points. For this reason the solution rank of the MVU problem seems to depend somewhat arbitrarily on n depending on the random sampling of the dataset and possible sampling artifacts.

The gap between the principal singular values representing the intrinsic dimensions of the input data and the remaining ones suggests an idea for obtaining approximate solutions of the MVU problem (2) at a lower computational cost. Namely, if we know the intrinsic dimension of the data a priori, we can solve problem (1) just once by setting d directly to this value. Unfortunately, the theoretical results of Section 2 do not guarantee that such solutions are either global solutions of (1) or solutions of the SDP (2). However, in our experience such solutions are usually satisfactory for visualization purposes. This ability to exploit a priori information about the intrinsic dimension is unique to the low-rank approach, and it is not available when using the standard SDP solvers.

5 Conclusions

The graph embedding problem (1) is relevant for many practical applications. For instance, maximum variance unfolding (MVU) is a well-known dimensionality reduction method that solves a special case of problem (1). The standard approach to solving problem (1) is to reformulate it as the semidefinite program (2) and use an interior-point SDP solver. This approach is not, however, computationally tractable for larger datasets (say $n > 2500$).

We developed an incremental low-rank method that solves a sequence of smaller quadratic problems (1) with increasing dimension d instead of solving the SDP (2) directly. Utilizing the existing theory on the relation between semidefinite programs and their quadratic low-rank reformulations, we rigorously guaranteed that after a finite number of steps, the low-rank method yields a solution to the SDP. Furthermore, such a solution is a global solution to the QCQP (1) with dimension d^* that gives an upper bound for the rank of the optimal solution of the SDP. We also derived a necessary and sufficient condition for testing these properties. Though not guaranteed by the theoretical results, our numerical experiments show that in practice it suffices to solve the QCQP only up to dimension that is the rank of the optimal solution to the SDP. This observation makes the incremental low-

rank approach particularly efficient when the input graph has a low-dimensional embedding.

In our numerical experiments, we considered the MVU problem that is a special case of the graph embedding SDP (2). We demonstrated that for solving this problem, the incremental low-rank approach coupled with the `IPOPT` solver is highly efficient. We were able to solve the MVU problem for three test datasets with up to 75000 points in less than an hour on an average workstation, which is by no means possible with the previously proposed methods. Our method is not only more efficient than the interior-point SDP solvers but also more efficient than `SDPLR`, an earlier low-rank method that uses a first-order L-BFGS algorithm for solving the quadratic subproblems.

Since we showed the gap between the primal and dual solutions of the SDP (2) being zero under mild assumptions, by solving the primal problem we obtain an optimal solution to the dual problem as well. The dual problem is equivalent to determining the fastest mixing Markov process on a graph (see e.g. [24]), and similar problems also appear in graph theory (see e.g. [12]). Thus, our low-rank approach is likely to be computationally efficient for many optimization problems that can be formulated as the dual of the graph embedding problem.

An important topic of future research is to incorporate the noise reduction method developed in [21] as a preprocessing step for MVU. Combined with the low-rank approach proposed in this paper, such a method, being scalable to large datasets and tolerant to noise, would address two of the most important shortcomings of the original MVU method. However, both in [21] and this paper we have considered only low-dimensional datasets. Efficient implementation of the density estimation method of [21] and finding the k-nearest neighbours of each data point in the MVU method for high-dimensional point sets is a very challenging problem.

Acknowledgements. The author was financially supported by the TUCS Graduate Programme. The author would also like to thank Prof. Marko M. Mäkelä and Doc. Napsu Karmitsa for their guidance in preparing this report and Paavo Nevalainen for implementing an efficient k-nearest neighbour algorithm.

References

[1] HSL (2011). A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk.

[2] P.R. Amestoy, I.S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184(2–4):501–520, 2000.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, third edition, 1999.

[4] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming - Theory and Algorithms*. John Wiley & Sons, Inc., New York, third edition, 2006.

[5] S.J. Benson and Y. Ye. Algorithm 875: DSDP5—software for semidefinite programming. *ACM Transactions on Mathematical Software (TOMS)*, 34(3):16:1–16:20, 2008.

[6] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1–4):613–623, 1999.

[7] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

[8] S. Burer and C. Choi. Computational enhancements in low-rank semidefinite programming. *Optimization Methods and Software*, 21(3):493–512, 2006.

[9] S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[10] S. Burer and R.D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[11] L. Grippo, L. Palagi, and V. Piccialli. Necessary and sufficient global optimality conditions for NLP reformulations of linear SDP problems. *Journal of Global Optimization*, 44(3):339–348, 2009.

[12] F. Göring, C. Helmberg, and M. Wappler. Embedded in the shadow of the separator. *SIAM Journal on Optimization*, 19(1):472–501, 2008.

[13] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996.

[14] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[15] M. Journée, F. Bach, P. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[16] B. Kulis, A.C. Surendran, and J.C. Platt. Fast low-rank semidefinite programming for embedding and clustering. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2007*, pages 512–521, 2007.

[17] R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998.

[18] G. Meurant. *The Lanczos and Conjugate Gradient Algorithms - From Theory to Finite Precision Computations*. SIAM, Philadelphia, 2006.

[19] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, second edition, 2006.

[20] J. Nocedal, A. Wächter, and R. Waltz. Adaptive barrier update strategies for nonlinear interior methods. *SIAM Journal on Optimization*, 19(4):1674–1693, 2009.

[21] S. Pulkkinen, M. M. Mäkelä, and N. Karmitsa. A generalized trust region Newton method applied to noise reduction. TUCS Technical Report TR1061, 2012.

[22] L.K. Saul, K.Q. Weinberger, J.H. Ham, F. Sha, and D.D. Lee. Spectral methods for dimensionality reduction. In B. Schölkopf, O. Chapelle, and A. Zien, editors, *Semisupervised Learning*, pages 293–308. The MIT Press, Cambridge, MA, USA, 2006.

[23] J.F. Sturm. Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1–4):625–653, 1999.

[24] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 48(4):681–699, 2006.

[25] K.C. Toh, M.J. Todd, and R.H. Tütüncü. SDPT3 — A MATLAB software package for semidefinite programming, Version 1.3. *Optimization Methods and Software*, 11(1–4):545–581, 1999.

[26] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[27] N. Vasiloglou, A.G. Gray, and D.V. Anderson. Scalable semidefinite manifold learning. In *IEEE Workshop on Machine Learning for Signal Processing, MLSP 2008*, pages 368–373, 2008.

[28] K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[29] R.C. Whaley, A. Petitet, and J.J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1-–2):3–35, 2001.

[30] A. Wächter and L.T. Biegler. Line search filter methods for nonlinear programming: motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

[31] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[32] L. Xiao, J. Sun, and S. Boyd. A duality view of spectral methods for dimensionality reduction. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 1041–1048, 2006.

[33] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of SDPA 6.0 (Semidefinite Programming Algorithm 6.0). *Optimization Methods and Software*, 18(4):491–505, 2003.

A Strong Duality of the Graph Embedding SDP

In this appendix we establish *strong duality* for primal-dual solution pairs of the graph embedding SDP (2). That is, we show that the primal problem and its dual have nonempty solution sets and their objective function values at an optimal solution pair coincide. This property, when it holds, is called *zero duality gap*. For the dual of problem (2), we use the formulation given by Sun et al. [24] and Xiao et al. [32]. They show that the dual problem can be written as

$$\begin{aligned} \min_{\boldsymbol{\lambda} \geq \mathbf{0}} \quad & \boldsymbol{b}^T \boldsymbol{\lambda} \\ \text{s.t.} \quad & \kappa(\sum_{i=1}^{n_E} \lambda_i \boldsymbol{A}_i) \geq 1, \end{aligned} \tag{27}$$

where n_E denotes the number of edges in the edge set E of the graph G, the function $\kappa(\cdot)$ denotes the second smallest eigenvalue of a matrix and the matrices $\boldsymbol{A}_i$ are defined according to equations (12) and (13). As in Subsection 2.2, we assume that the edges of the graph G can be ordered according to (11) and define the vector $\boldsymbol{b}$ as $b_k = D^2_{i_k,j_k}$ for $k = 1, 2, \ldots, n_E$.

We will establish our main results under two mild assumptions on the input graph G. The first assumption is a nondegeneracy condition that excludes zero edge lengths.

Assumption A.1. *The edge lengths D_{ij} of the graph $G = (V, E)$ satisfy the condition $D_{ij} > 0$ for all $\{i, j\} \in E$.*

The second assumption is connectedness of the graph. For the MVU problem, this can be in practice guaranteed either by choosing a sufficiently large neighbourhood size k or by considering distinct components of the graph separately.

Assumption A.2. *The graph $G = (V, E)$ is connected. That is, for any pair $\{i, j\} \subset V$ there exists a sequence of indices $k_1, k_2, \ldots, k_m$ such that $k_1 = i$, $k_m = j$ and $\{k_i, k_{i+1}\} \in E$ for all $i = 1, 2, \ldots, m-1$.*

Next, we show that problem (2) has a bounded and nonempty feasible set under Assumption A.2.

Theorem A.1. *Under Assumption A.2, the feasible set of problem (2) is bounded and nonempty.*

Proof. Let $\boldsymbol{K} \in \mathcal{S}^n$ be a feasible matrix for problem (2) and let us denote by

$$\tau = \max_{\{i,j\} \in E} D_{ij}$$

the maximum edge length of the graph G. By condition (2d) we obtain that

$$\operatorname{tr}(\boldsymbol{K}) = \sum_{i=1}^{n} K_{ii} = \frac{1}{2n} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} K_{ii} - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} K_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} K_{jj} \right). \tag{28}$$

On the other hand, as a positive semidefinite matrix $\boldsymbol{K}$ admits the factorization $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^T$ with some $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1, & \boldsymbol{y}_2, & \ldots, & \boldsymbol{y}_n \end{bmatrix}^T \in \mathbb{R}^{n \times d}$ such that $d \leq n$. By Assumption A.2, for all $i, j \in \{1, 2, \ldots, n\}$, the triangular inequality and conditions (2c) we then obtain

$$\begin{aligned} K_{ii} - 2K_{ij} + K_{jj} &= \boldsymbol{y}_i^T \boldsymbol{y}_i - 2\boldsymbol{y}_i^T \boldsymbol{y}_j + \boldsymbol{y}_j^T \boldsymbol{y}_j \\ &= \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \leq \left(\sum_{l=1}^{m-1} \|\boldsymbol{y}_{k_{l+1}} - \boldsymbol{y}_{k_l}\| \right)^2 \leq n^2 \tau^2 \end{aligned}$$

with some sequence $\{k_l\}$ satisfying the conditions stated in Assumption A.2. Substituting the above inequality into equation (28) yields the upper bound

$$\operatorname{tr}(\boldsymbol{K}) \leq \frac{n^3 \tau^2}{2}. \tag{29}$$

As a positive semidefinite matrix $\boldsymbol{K}$ satisfies the inequality

$$|K_{ij}| \leq \frac{K_{ii} + K_{jj}}{2} \leq \frac{\operatorname{tr}(\boldsymbol{K})}{2}, \quad i, j = 1, 2, \ldots, n$$

(see e.g. [14], p. 398), which together with inequality (29) shows that the elements of $\boldsymbol{K}$ are bounded. Finally, the feasible set of problem (2) is nonempty since the matrix $\boldsymbol{K} = \boldsymbol{0}$ trivially satisfies conditions (2b)–(2d). □

Since problem (14) is obtained from problem (15) by the variable substitution $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^T$ that yields a positive semidefinite matrix, any feasible matrix $\boldsymbol{Y}$ for problem (14) with any dimension d clearly yields a feasible matrix $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^T$ for problem (15). This implies that also the feasible set of (NLP$_d$), which is equivalent to (14), is bounded under Assumption A.2. The feasible set is also nonempty since for any dimension d, the zero vector $\boldsymbol{0}_{nd}$ is feasible. Hence, any solution algorithm applied to problem (NLP$_d$) in Algorithm 1 is guaranteed to give a bounded solution whenever Assumption A.2 is satisfied. Furthermore, these properties apply to problem (1) via Proposition 2.1.

Corollary A.1. *Under Assumption A.2, the feasible set of problems* (1)*,* (14) *and* (NLP$_d$) *is bounded for all d.*

By using Theorem A.1, we will now show that the feasible set of the dual problem (27) has nonempty interior under Assumption A.2.

Theorem A.2. *Under Assumption A.2, the feasible set of problem* (27) *has nonempty interior.*

Proof. The proof is obtained by contradiction. Let Assumption A.2 be satisfied and assume that the interior of the feasible set of problem (27) is empty. From definitions (12) and (13), we note that for any choice of $\boldsymbol{\lambda} \geq \boldsymbol{0}$ the matrix $\sum_{i=1}^{n_E} \lambda_i \boldsymbol{A}_i$ is positive semidefinite and its smallest eigenvalue is zero with eigenvector $\boldsymbol{1}_n$. Thus, it suffices to consider eigenvectors lying in the subspace

$$V = \{\boldsymbol{v} \in \mathbb{R}^n \mid \|\boldsymbol{v}\| = 1, \boldsymbol{v}^T \boldsymbol{1}_n = 0\}.$$

Then by our assumption about emptiness of the interior of the feasible set of problem (27), for all $\boldsymbol{\lambda} \geq \mathbf{0}$ there exists a vector $\boldsymbol{v} \in V$ such that

$$\boldsymbol{v}^T(\sum_{i=1}^{n_E} \lambda_i \boldsymbol{A}_i)\boldsymbol{v} = \boldsymbol{v}^T(\sum_{i=1}^{n_E} \lambda_i \boldsymbol{a}_i \boldsymbol{a}_i^T)\boldsymbol{v} = \sum_{i=1}^{n_E} \lambda_i (\boldsymbol{v}^T \boldsymbol{a}_i)^2 \leq 1.$$

Since the above inequality holds for $\boldsymbol{\lambda} = \alpha \mathbf{1}_{n_E}$ for any $\alpha > 0$, we deduce that for all $\alpha > 0$ there exists a vector $\boldsymbol{v} \in V$ such that

$$\boldsymbol{v}^T(\sum_{i=1}^{n_E} \boldsymbol{A}_i)\boldsymbol{v} = \sum_{i=1}^{n_E} (\boldsymbol{v}^T \boldsymbol{a}_i)^2 \leq \frac{1}{\alpha}.$$

Consequently, since α is unbounded from above and the set V is compact, this implies that there exists $\boldsymbol{v} \in V$ such that $\boldsymbol{v}^T \boldsymbol{a}_i = 0$ for all $i = 1, 2, \ldots, n_E$. Let us then define the matrix $\boldsymbol{K}(\alpha) = \alpha \boldsymbol{v}\boldsymbol{v}^T$ with such a vector $\boldsymbol{v}$. Then by expressing constraints (2c) in terms of the vectors $\boldsymbol{a}_i$, using the identity

$$(\boldsymbol{a}\boldsymbol{a}^T) \bullet (\boldsymbol{v}\boldsymbol{v}^T) = (\boldsymbol{a}^T \boldsymbol{v})^2$$

for any vectors $\boldsymbol{a}, \boldsymbol{v} \in \mathbb{R}^n$ and the property that $\boldsymbol{a}_i^T \mathbf{1}_n = 0$ for all $i = 1, 2, \ldots, n_E$ we obtain that

$$\begin{aligned} K_{i_k,i_k}(\alpha) - 2K_{i_k,j_k}(\alpha) + K_{j_k,j_k}(\alpha) &= (\boldsymbol{a}_k \boldsymbol{a}_k^T) \bullet \boldsymbol{K}(\alpha) \\ &= (\boldsymbol{a}_k \boldsymbol{a}_k^T) \bullet (\alpha \boldsymbol{v}\boldsymbol{v}^T) \\ &= \alpha (\boldsymbol{a}_k^T \boldsymbol{v})^2 = 0 \end{aligned}$$

for all $k = 1, 2, \ldots, n_E$. Thus, the matrix $\boldsymbol{K}(\alpha)$ satisfies conditions (2c). Furthermore, by the definition of $\boldsymbol{K}(\alpha)$ and the fact that $\boldsymbol{v} \in V$ we have

$$\sum_{i=1}^{n} \sum_{j=1}^{n} K_{ij}(\alpha) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha v_i v_j = \sum_{i=1}^{n} \alpha v_i \boldsymbol{v}^T \mathbf{1}_n = 0,$$

and thus $\boldsymbol{K}(\alpha)$ satisfies condition (2d). In addition, the matrix $\boldsymbol{K}(\alpha)$ is clearly positive semidefinite, which together with the above observations implies that it is feasible for problem (2) with any $\alpha \geq 0$. This leads to contradiction with the result of Theorem A.1, which states that under Assumption A.2 the feasible set of problem (2) is bounded. □

We shall now show that problem (2) and its dual (27) have nonempty solution sets and their objective function values coincide at these optimal solutions. In order to show this via the standard duality results, we need to transform problem (2) into a problem whose feasible set has nonempty relative interior (i.e. there exists a feasible $\boldsymbol{K}$ such that $\boldsymbol{K} \succ \mathbf{0}$). As shown in [24], problem (2) can be reformulated as

$$\begin{aligned} \max_{\boldsymbol{K} \in \mathcal{S}^n} \quad & \text{tr}[(\boldsymbol{I} - \frac{1}{n}\mathbf{1}_n \mathbf{1}_n^T)\boldsymbol{K}] \\ \text{s.t.} \quad & \boldsymbol{K} \succeq 0, \\ & K_{ii} - 2K_{ij} + K_{jj} \leq D_{ij}^2, \quad \{i,j\} \in E \end{aligned} \tag{30}$$

by lifting the constraint (2d) into the objective function.

Remark A.1. *The feasible set of problem* (2) *does not have a nonempty relative interior. Namely, any positive semidefinite matrix* $\boldsymbol{K}$ *can be factorized as* $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^T$ *for some* $\boldsymbol{Y} \in \mathbb{R}^d$*. This implies that if* $\boldsymbol{K}$ *satisfies condition* (2d)*, then condition* (14c) *holds. By using the properties of the operator* $\bullet$*, we then obtain that* $(\mathbf{1}_n\mathbf{1}_n^T)\bullet(\boldsymbol{Y}\boldsymbol{Y}^T) = \|\mathbf{1}^T\boldsymbol{Y}\|^2 = 0$*. This implies that* $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^T$ *has eigenvalue zero with eigenvector* $\mathbf{1}_n$*, and thus* $\boldsymbol{K}$ *cannot be positive definite.*

Remark A.2. *Problem* (30) *has the same dual problem with problem* (2)*, that is problem* (27)*. Furthermore, it can be shown that the solution set of problem* (2) *is in the solution set of problem* (30)*, and the objective function values of these two problems coincide at a solution that is optimal for both problems. Thus, by showing that the primal problem* (30) *and its dual* (27) *have zero gap, we also establish this property for problems* (2) *and* (27)*.*

For problem (30) we can show that its feasible set has nonempty relative interior.

Theorem A.3. *Under Assumption A.1, the feasible set of problem* (30) *has nonempty relative interior.*

Proof. Let us define the matrix $\boldsymbol{K} = \alpha\boldsymbol{I}$, where $\alpha \in]0, \beta]$ with

$$\beta = \frac{1}{2}\min_{\{i,j\}\in E} D_{ij}^2.$$

By Assumption A.1 such choice is possible. Clearly, the matrix $\boldsymbol{K}$ is positive definite (i.e. $\boldsymbol{K} \succ \mathbf{0}$) and satisfies the condition $K_{ii} - 2K_{ij} + K_{jj} \leq D_{ij}^2$ for all $\{i, j\} \in E$. □

Finally, by using Theorems A.2 and A.3 and Remark A.2, we can show that the gap between the solutions of the primal problem (2) and its dual (27) is zero. This guarantees that Assumption 2.1 is satisfied by the primal problem (2) and its dual once the primal problem has been transformed into the standard form (4).

Theorem A.4. *Under Assumptions A.1 and A.2, the primal problem* (2) *and its dual* (27) *have nonempty solution sets. Furthermore, if* $\boldsymbol{K}^* \in \mathbb{R}^{n\times n}$ *is a solution to* (2) *and* $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$ *is a solution to* (27)*, then* $\mathrm{tr}(\boldsymbol{K}^*) = \boldsymbol{b}^T\boldsymbol{\lambda}^*$*.*

Proof. By virtue of Theorems A.2 and A.3, the feasible sets of problems (30) and (27) have nonempty relative interior. Furthermore, problem (30) is convex since it has a linear objective function and linear constraints and the convex nonlinear constraint $\boldsymbol{K} \succeq \mathbf{0}$. Convexity of the positive semidefiniteness constraint $\boldsymbol{K} \succeq \mathbf{0}$ is shown, for instance, in [26]. On the other hand, convexity of problem (27) follows from the fact that its objective function is linear and the second smallest eigenvalue of the matrix $\sum_{i=1}^{n_E} \lambda_i \boldsymbol{A}_i$ is a concave function of $\boldsymbol{\lambda}$ (see e.g. [24] for concavity of this function).

Hence, we have shown that problems (30) and its dual (27) are convex and their feasible sets have nonempty relative interior. The claim for the original problem (2) and its dual (27) then follows from Remark A.2 and the fact that under these conditions the primal and dual problems have nonempty solution sets and the objective function values of the primal and dual solutions coincide (see e.g. [7] and [26]). □

B Proofs of Technical Results

In this appendix we prove some technical results used in Sections 2 and 3. We begin with the proof of Proposition 2.1.

Proof of Proposition 2.1. First, we observe that by condition (1b) all feasible points $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathbb{R}^d$ of problem (1) satisfy

$$\sum_{i=1}^n \sum_{j=1}^n \boldsymbol{y}_i^T \boldsymbol{y}_j = \| \sum_{i=1}^n \boldsymbol{y}_i \|^2 = 0.$$

Thus, the objective function of problem (1) can be equivalently stated as

$$\begin{aligned}\sum_{i=1}^n \sum_{j=1}^n \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 &= \sum_{i=1}^n \sum_{j=1}^n \|\boldsymbol{y}_i\|^2 - 2 \sum_{i,j=1}^n \boldsymbol{y}_i^T \boldsymbol{y}_j + \sum_{i=1}^n \sum_{j=1}^n \|\boldsymbol{y}_j\|^2 \\ &= 2n \sum_{i=1}^n \|\boldsymbol{y}_i\|^2 = 2n \cdot \mathrm{tr}(\boldsymbol{Y}\boldsymbol{Y}^T) = -2n \cdot \boldsymbol{C} \bullet (\boldsymbol{Y}\boldsymbol{Y}^T)\end{aligned}$$

with $\boldsymbol{C} = -\boldsymbol{I}_n$.

On the other hand, by equations (12) and (13) and the properties of the operator $\bullet$ we obtain that

$$\boldsymbol{A}_k \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) = (\boldsymbol{a}_k \boldsymbol{a}_k^T) \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) = \|\boldsymbol{a}_k^T \boldsymbol{Y}\|^2$$

and

$$\boldsymbol{a}_k^T \boldsymbol{Y} = [y_{i_k,1} - y_{j_k,1}, \quad y_{i_k,2} - y_{j_k,2}, \quad \ldots, \quad y_{i_k,d} - y_{j_k,d}] \in \mathbb{R}^d,$$

and thus

$$\boldsymbol{A}_k \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) = \|\boldsymbol{y}_{i_k} - \boldsymbol{y}_{j_k}\|^2, \quad k = 1, 2, \ldots, n_E.$$

Finally, we observe that the condition

$$(\mathbf{1}_n \mathbf{1}_n^T) \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) = \sum_{i=1}^n \sum_{j=1}^n \boldsymbol{y}_i^T \boldsymbol{y}_j = \| \sum_{i=1}^n \boldsymbol{y}_i \|^2 = \sum_{j=1}^d \| \sum_{i=1}^n y_{i,j} \|^2 = 0$$

is equivalent to the condition that $\sum_{i=1}^n y_{i,j} = 0$ for all $j = 1, 2, \ldots, d$. This in turn is equivalent to condition (1b). □

For the proof of Theorem 3.1, we use a technical lemma for the optimal Lagrange multipliers of problem (14).

Lemma B.1. *If $\boldsymbol{Y}^* \in \mathbb{R}^{n \times d}$ is a solution to (14) with Lagrange multiplier μ^* corresponding to the constraint (14c), then $\mu^* = \frac{1}{n}$.*

Proof. The Lagrangian of problem (14) is given by

$$\mathcal{L}(\boldsymbol{Y}; \boldsymbol{\lambda}; \mu) = -\mathrm{tr}(\boldsymbol{Y}\boldsymbol{Y}^T) + \sum_{i=1}^{n_E} [\lambda_i (\boldsymbol{A}_i \bullet (\boldsymbol{Y}\boldsymbol{Y}^T) - b_i] + \mu (\mathbf{1}_n \mathbf{1}_n^T) \bullet (\boldsymbol{Y}\boldsymbol{Y}^T).$$

Let $\boldsymbol{Y}^* \in \mathbb{R}^{n\times d}$ be a solution to problem (14) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$ and $\mu^* \in \mathbb{R}$. This implies the first-order necessary KKT condition

$$\nabla_{\boldsymbol{Y}} \mathcal{L}(\boldsymbol{Y}^*; \boldsymbol{\lambda}^*; \mu^*) = -2\boldsymbol{Y}^* + 2\sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i \boldsymbol{Y}^* + 2\mu^* \mathbf{1}_n \mathbf{1}_n^T \boldsymbol{Y}^* = \mathbf{0}.$$

Premultiplying this condition by $\boldsymbol{Y}^{*^T} \mathbf{1}_n \mathbf{1}_n^T$ yields

$$n\mu^* \boldsymbol{Y}^{*^T} \mathbf{1}_n \mathbf{1}_n^T \boldsymbol{Y}^* = \boldsymbol{Y}^{*^T} \mathbf{1}_n \mathbf{1}_n^T \boldsymbol{Y}^*$$

by noting that $\mathbf{1}_n^T \boldsymbol{A}_i = \mathbf{1}_n^T \boldsymbol{a}_i \boldsymbol{a}_i^T = \mathbf{0}$ for all $i = 1, 2, \ldots, n_E$ and that $\mathbf{1}_n^T \mathbf{1}_n = n$. This is equivalent to the condition that $\mu^* = \frac{1}{n}$. □

Proof of Theorem 3.1. Assume first that $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a solution to problem (NLP$_d$) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$ and

$$\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) = \kappa(\sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i) \geq 1.$$

By the elementary property of eigenvalues that

$$\kappa(\alpha \boldsymbol{I} + \boldsymbol{A}) = \alpha + \kappa(\boldsymbol{A}), \tag{31}$$

for any $\alpha \in \mathbb{R}$ this implies that

$$\kappa(-\boldsymbol{I}_n + \sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i) \geq 0. \tag{32}$$

On the other hand, we have the identities

$$(\sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i)\mathbf{1}_n = \mathbf{0} \quad \text{and} \quad (-\boldsymbol{I}_n + \frac{1}{n}\mathbf{1}_n \mathbf{1}_n^T)\mathbf{1}_n = \mathbf{0}, \tag{33}$$

where the first identity follows from equations (12) and (13). These identities imply that the eigenvector $\mathbf{1}_n$ corresponds to the zero eigenvalue of the matrix

$$-\boldsymbol{I}_n + \sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i + \frac{1}{n}\mathbf{1}_n \mathbf{1}_n^T, \tag{34}$$

and hence this matrix is positive semidefinite by inequality (32). By the assumption that $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a solution to problem (NLP$_d$), the matrix $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$ is a solution to the equivalent problem (14). Since by Lemma B.1 the Lagrange multiplier μ^* corresponding to the constraint (14c) is $\frac{1}{n}$, Theorem 2.1 applied to problems (14) and (15) then implies that $\boldsymbol{K}^* = \boldsymbol{Y}^* \boldsymbol{Y}^{*^T}$ with $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$ is a solution to (2) and $\boldsymbol{y}^*$ is a global solution to (NLP$_d$).

Conversely, assume then that $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a global solution to (NLP$_d$) and the matrix $\boldsymbol{K}^* = \boldsymbol{Y}^* \boldsymbol{Y}^{*^T}$ with $\boldsymbol{Y}^* = \text{mat}(\boldsymbol{y}^*)$ is a solution to (2). Then Propositions 2.1 and 2.2

imply that $\boldsymbol{Y}^*$ is a solution to (14) and $\boldsymbol{K}^*$ is a solution to (15). Consequently, Theorem 2.2 applied to problems (14) and (15) implies that the condition

$$-\boldsymbol{I}_n + \sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i + \mu^* (\mathbf{1}_n \mathbf{1}_n^T) \succeq \mathbf{0}$$

holds with the optimal Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$ and $\mu^* \in \mathbb{R}$. On the other hand, by Lemma B.1 we have $\mu^* = \frac{1}{n}$. Thus, the above condition and identities (33) imply that

$$\kappa(-\boldsymbol{I}_n + \sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i) \geq 0$$

since the remaining eigenvectors of the symmetric matrix (34) are orthogonal to the eigenvector $\mathbf{1}_n$. By equation (31), this in turn implies that

$$\kappa(\sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i) = \kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) \geq 1.$$

□

Finally, we prove Theorem 3.2 by using a technical lemma for the optimal Lagrange multipliers of problem (NLP$_d$).

Lemma B.2. *If $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ is a first-order KKT point of problem (NLP$_d$) with Lagrange multipliers $\boldsymbol{\mu}^* \in \mathbb{R}^d$ corresponding to the constraints $h_i^d(\boldsymbol{y}^*) = 0$, then $\boldsymbol{\mu}^* = \mathbf{0}$.*

Proof. Let $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ be a first-order KKT point of problem (NLP$_d$) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$ and $\boldsymbol{\mu}^* \in \mathbb{R}^d$. This implies the first-order necessary KKT condition

$$\nabla_{\boldsymbol{y}} \mathcal{L}_d(\boldsymbol{y}^*; \boldsymbol{\lambda}^*; \boldsymbol{\mu}^*) = -2\boldsymbol{y}^* + 2\sum_{i=1}^{n_E} \lambda_i^* (\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y}^* + \sum_{i=1}^{d} \mu_i^* \boldsymbol{c}_i^d = \mathbf{0}. \tag{35}$$

On the other hand, we note that

$$(\boldsymbol{c}_j^d)^T (\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y}^* = (\boldsymbol{c}_j^d)^T (\boldsymbol{I}_d \otimes \boldsymbol{a}_i)(\boldsymbol{I}_d \otimes \boldsymbol{a}_i)^T \boldsymbol{y}^* = 0$$

for all $j = 1, 2, \ldots, d$. This is because we have $(\boldsymbol{I}_d \otimes \boldsymbol{a}_i)^T \boldsymbol{c}_j^d = \mathbf{0}$ for all $i = 1, 2, \ldots, n_E$ and $j = 1, 2, \ldots, d$ by equation (18) and the fact that $\mathbf{1}_n^T \boldsymbol{a}_i = 0$ for all $i = 1, 2, \ldots, n_E$. On the other hand, we have $(\boldsymbol{c}_j^d)^T \boldsymbol{y}^* = 0$ by the feasibility of $\boldsymbol{y}^*$ for problem (NLP$_d$). By equation (18), premultiplying equation (35) by $(\boldsymbol{c}_j^d)^T$ for any $j = 1, 2, \ldots, d$ then yields

$$-2(\boldsymbol{c}_j^d)^T \boldsymbol{y}^* + 2\sum_{i=1}^{n_E} \lambda_i^* (\boldsymbol{c}_j^d)^T (\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y}^* + \sum_{i=1}^{d} \mu_i^* (\boldsymbol{c}_j^d)^T \boldsymbol{c}_i^d = \begin{cases} n\mu_i^*, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases}$$

This shows that the first-order necessary KKT condition (35) can only be satisfied when $\mu_i^* = 0$ for all $i = 1, 2, \ldots, d$. □

Proof of Theorem 3.2. First, we note that the Lagrangian of problem (NLP$_d$) with a given dimension d has gradient

$$\nabla_{\boldsymbol{y}} \mathcal{L}_d(\boldsymbol{y}; \boldsymbol{\lambda}; \boldsymbol{\mu}) = -2\boldsymbol{y} + 2\sum_{i=1}^{n_E} \lambda_i (\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y} + \sum_{i=1}^{d} \mu_i^* \boldsymbol{c}_i^d$$

and Hessian

$$\nabla_{\boldsymbol{y}}^2 \mathcal{L}_d(\boldsymbol{y}; \boldsymbol{\lambda}; \boldsymbol{\mu}) = -2\boldsymbol{I}_{nd} + 2\sum_{i=1}^{n_E} \lambda_i (\boldsymbol{I}_d \otimes \boldsymbol{A}_i).$$

Then, let $\boldsymbol{y}^* \in \mathbb{R}^{nd}$ be a first-order KKT point of problem (NLP$_d$) with Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_E}$, $\boldsymbol{\lambda}^* \geq \boldsymbol{0}$, and $\boldsymbol{\mu}^* \in \mathbb{R}^d$ such that condition (20) is not satisfied.

The vector $\tilde{\boldsymbol{y}}^*$ is feasible for problem (NLP$_{d+1}$) since clearly

$$g_i^{d+1}(\tilde{\boldsymbol{y}}^*) = \tilde{\boldsymbol{y}}^{*^T} (\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i)\tilde{\boldsymbol{y}}^* = \boldsymbol{y}^{*^T} (\boldsymbol{I}_d \otimes \boldsymbol{A}_i)\boldsymbol{y}^* = g_i^d(\boldsymbol{y}^*) \leq b_i$$

for all $i = 1, 2, \ldots, n_E$,

$$h_i^{d+1}(\tilde{\boldsymbol{y}}^*) = (\boldsymbol{c}_i^{d+1})^T \tilde{\boldsymbol{y}}^* = (\boldsymbol{c}_i^d)^T \boldsymbol{y}^* = h_i^d(\boldsymbol{y}^*) = 0$$

for all $i = 1, 2, \ldots, d$ and

$$h_{d+1}^{d+1}(\tilde{\boldsymbol{y}}^*) = (\boldsymbol{c}_{d+1}^{d+1})^T \tilde{\boldsymbol{y}}^* = \boldsymbol{0}_{nd}^T \boldsymbol{y}^* + \boldsymbol{1}_n^T \boldsymbol{0}_n = 0.$$

Furthermore, by Lemma B.2 we have $\mu_i^* = 0$ for all $i = 1, 2, \ldots, d$. Consequently,

$$\nabla_{\boldsymbol{y}} \mathcal{L}_{d+1}(\tilde{\boldsymbol{y}}^*; \boldsymbol{\lambda}^*; \tilde{\boldsymbol{\mu}}^*) = -2\tilde{\boldsymbol{y}}^* + 2\sum_{i=1}^{n_E} \lambda_i^* (\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i)\tilde{\boldsymbol{y}}^* = \begin{bmatrix} \nabla_{\boldsymbol{y}} \mathcal{L}_d(\boldsymbol{y}^*; \boldsymbol{\lambda}^*; \boldsymbol{\mu}^*) \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{0},$$

which shows that $\tilde{\boldsymbol{y}}^*$ is a first-order KKT point of problem (NLP$_{d+1}$) with Lagrange multipliers $\boldsymbol{\lambda}^*$ and $\tilde{\boldsymbol{\mu}}^*$.

On the other hand, let $\boldsymbol{v}^* \in \mathbb{R}^n$ be an eigenvector corresponding to the eigenvalue

$$\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) = \kappa(\sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i).$$

Without loss of generality, we can assume that $\|\boldsymbol{v}^*\| = 1$. Then, by the assumption that $\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) < 1$ we have

$$\boldsymbol{v}^{*^T} (\sum_{i=1}^{n_E} \lambda_i^* \boldsymbol{A}_i)\boldsymbol{v}^* < 1,$$

which by the definition of the Kronecker product $\otimes$ implies that

$$\boldsymbol{d}^T [\sum_{i=1}^{n_E} \lambda_i^* (\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i)]\boldsymbol{d} < 1$$

with

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{0}_{nd} \\ \boldsymbol{v}^* \end{bmatrix}.$$

This in turn implies that

$$\boldsymbol{d}^T[-\boldsymbol{I}_{n(d+1)} + \sum_{i=1}^{n_E} \lambda_i^*(\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i)]\boldsymbol{d} = \frac{1}{2}\boldsymbol{d}^T \nabla_{\boldsymbol{y}}^2 \mathcal{L}_{d+1}(\tilde{\boldsymbol{y}}^*; \boldsymbol{\lambda}^*; \tilde{\boldsymbol{\mu}}^*)\boldsymbol{d} < 0.$$

Finally, for the constraint gradients of problem (NLP$_{d+1}$) we obtain

$$\begin{aligned}\nabla g_i^{d+1}(\tilde{\boldsymbol{y}}^*)^T\boldsymbol{d} = 2\tilde{\boldsymbol{y}}^{*T}[\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i]\boldsymbol{d} &= 2\begin{bmatrix}\boldsymbol{y}^{*T} & \boldsymbol{0}_n^T\end{bmatrix}[\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i]\begin{bmatrix}\boldsymbol{0}_{nd} \\ \boldsymbol{v}^*\end{bmatrix} \\ &= 2\begin{bmatrix}\boldsymbol{y}^{*T} & \boldsymbol{0}_n^T\end{bmatrix}\begin{bmatrix}\boldsymbol{0}_{nd} \\ (\boldsymbol{I}_{d+1} \otimes \boldsymbol{A}_i)\boldsymbol{v}^*\end{bmatrix} = 0\end{aligned}$$

for $i = 1, 2, \ldots, n_E$ and similarly

$$\nabla h_i^{d+1}(\tilde{\boldsymbol{y}}^*)^T\boldsymbol{d} = (\boldsymbol{c}_i^{d+1})^T\begin{bmatrix}\boldsymbol{0}_{nd} \\ \boldsymbol{v}^*\end{bmatrix} = 0$$

for $i = 1, 2, \ldots, d$. Finally, assuming that $\boldsymbol{1}_n^T\boldsymbol{v}^* = 0$, we have

$$\nabla h_{d+1}^{d+1}(\tilde{\boldsymbol{y}}^*)^T\boldsymbol{d} = (\boldsymbol{c}_{d+1}^{d+1})^T\begin{bmatrix}\boldsymbol{0}_{nd} \\ \boldsymbol{v}^*\end{bmatrix} = \begin{bmatrix}\boldsymbol{0}_{nd}^T & \boldsymbol{1}_n^T\end{bmatrix}\begin{bmatrix}\boldsymbol{0}_{nd} \\ \boldsymbol{v}^*\end{bmatrix} = \boldsymbol{1}_n^T\boldsymbol{v}^* = 0.$$

It is always possible to choose an eigenvector $\boldsymbol{v}^*$ satisfying the condition $\boldsymbol{1}_n^T\boldsymbol{v}^* = 0$. Namely, if $\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) \neq 0$, then due to orthogonality of the eigenvectors of the symmetric matrix $\boldsymbol{L}(\boldsymbol{\lambda}^*)$, any eigenvector in the eigenspace $\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*))$ is orthogonal to the vector $\boldsymbol{1}_n$. Namely, by equations (12) and (13) the vector $\boldsymbol{1}_n$ is an eigenvector of $\boldsymbol{L}(\boldsymbol{\lambda}^*)$ corresponding to the eigenvalue zero. Otherwise, if $\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*)) = 0$, by the orthogonality of the eigenvectors the eigenspace of $\kappa(\boldsymbol{L}(\boldsymbol{\lambda}^*))$ contains an eigenvector that is orthogonal to the eigenvector $\boldsymbol{1}_n$. □

C Equality-Constrained Formulation With Slack Variables

Semidefinite programs with inequality constraints can be equivalently transformed into the standard form SDP (4), and similarly, their low-rank formulations can be equivalently transformed into the standard form (3). To this end, we define the matrices

$$\tilde{\boldsymbol{C}} = \begin{bmatrix}\boldsymbol{C} & \boldsymbol{0}_{n\times m_1} \\ \boldsymbol{0}_{m_1\times n} & \boldsymbol{0}_{m_1\times m_1}\end{bmatrix}, \tag{36}$$

$$\tilde{\boldsymbol{A}}_i = \begin{bmatrix}\boldsymbol{A}_i & \boldsymbol{0}_{n\times m_1} \\ \boldsymbol{0}_{m_1\times n} & \boldsymbol{e}_i\boldsymbol{e}_i^T\end{bmatrix}, \qquad i = 1, 2, \ldots, m_1 \tag{37}$$

$$\tilde{\boldsymbol{B}}_i = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{0}_{n\times m_1} \\ \boldsymbol{0}_{m_1\times n} & \boldsymbol{0}_{m_1\times m_1}\end{bmatrix}, \qquad i = 1, 2, \ldots, m_2, \tag{38}$$

where $\boldsymbol{e}_i$ denotes a unit vector along the i-th coordinate axis. With these definitions, a straightforward calculation shows that for any feasible matrix $\boldsymbol{K}$ for the problem

$$\begin{aligned} \min_{\boldsymbol{K}\in\mathbb{R}^{n\times n}} \ & \boldsymbol{C}\bullet\boldsymbol{K} \\ \text{s.t. } & \boldsymbol{K}\succeq 0 \\ & \boldsymbol{A}_i\bullet\boldsymbol{K}\le b_i, \quad i=1,2,\ldots,m_1, \\ & \boldsymbol{B}_i\bullet\boldsymbol{K}= c_i, \quad i=1,2,\ldots,m_2 \end{aligned} \tag{39}$$

there exists an augmented matrix

$$\tilde{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{U} \\ \boldsymbol{U}^T & \boldsymbol{S} \end{bmatrix} \in \mathbb{R}^{(n+m_1)\times(n+m_1)}$$

with any $\boldsymbol{U}\in\mathbb{R}^{n\times m_1}$ and some $\boldsymbol{S}\in\mathbb{R}^{m_1\times m_1}$ that is feasible for the standard form problem

$$\begin{aligned} \min_{\tilde{\boldsymbol{K}}\in\mathbb{R}^{(n+m_1)\times(n+m_1)}} \ & \tilde{\boldsymbol{C}}\bullet\tilde{\boldsymbol{K}} \\ \text{s.t. } & \tilde{\boldsymbol{K}}\succeq 0 \\ & \tilde{\boldsymbol{A}}_i\bullet\tilde{\boldsymbol{K}}= b_i, \quad i=1,2,\ldots,m_1, \\ & \tilde{\boldsymbol{B}}_i\bullet\tilde{\boldsymbol{K}}= c_i, \quad i=1,2,\ldots,m_2 \end{aligned} \tag{40}$$

with the same objective and constraint function values. This equivalence also holds conversely.

Similarly, for any feasible matrix $\boldsymbol{Y}$ for the problem

$$\begin{aligned} \min_{\boldsymbol{Y}\in\mathbb{R}^{n\times d}} \ & \boldsymbol{C}\bullet(\boldsymbol{Y}\boldsymbol{Y}^T) \\ \text{s.t. } & \boldsymbol{A}_i\bullet(\boldsymbol{Y}\boldsymbol{Y}^T)\le b_i, \quad i=1,2,\ldots,m_1, \\ & \boldsymbol{B}_i\bullet(\boldsymbol{Y}\boldsymbol{Y}^T)= c_i, \quad i=1,2,\ldots,m_2 \end{aligned} \tag{41}$$

there exists an augmented matrix

$$\tilde{\boldsymbol{Y}} = \begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{S} \end{bmatrix} \in \mathbb{R}^{(n+m_1)\times d}$$

with some $\boldsymbol{S}\in\mathbb{R}^{m_1\times d}$ that is feasible for the standard form problem

$$\begin{aligned} \min_{\tilde{\boldsymbol{Y}}\in\mathbb{R}^{(n+m_1)\times d}} \ & \tilde{\boldsymbol{C}}\bullet(\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}^T) \\ \text{s.t. } & \tilde{\boldsymbol{A}}_i\bullet(\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}^T)= b_i, \quad i=1,2,\ldots,m_1, \\ & \tilde{\boldsymbol{B}}_i\bullet(\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}^T)= c_i, \quad i=1,2,\ldots,m_2 \end{aligned} \tag{42}$$

with the same objective and constraint function values. This equivalence also holds conversely.

In particular, the necessary and sufficient optimality condition (9) provided by Theorems 2.1 and 2.2 is directly applicable to the inequality-constrained problems (39) and (41). This follows from definitions (36)–(38) and the following well-known properties about eigenvalues and eigenvectors of block-diagonal matrices.

Lemma C.1. *Assume that the matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m\times m}$ have eigenvalues $\{\lambda_i\}_{i=1}^n$ and $\{\mu_i\}_{i=1}^m$ and eigenvectors $\{\boldsymbol{v}_i\}_{i=1}^n$ and $\{\boldsymbol{w}_i\}_{i=1}^m$, respectively. Then the matrix*

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} \end{bmatrix}$$

has eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n, \mu_1, \mu_2, \ldots, \mu_m\}$ and eigenvectors

$$\left\{ \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{0}_m \end{bmatrix}, \begin{bmatrix} \boldsymbol{v}_2 \\ \boldsymbol{0}_m \end{bmatrix}, \ldots, \begin{bmatrix} \boldsymbol{v}_n \\ \boldsymbol{0}_m \end{bmatrix}, \begin{bmatrix} \boldsymbol{0}_n \\ \boldsymbol{w}_1 \end{bmatrix}, \begin{bmatrix} \boldsymbol{0}_n \\ \boldsymbol{w}_2 \end{bmatrix}, \ldots, \begin{bmatrix} \boldsymbol{0}_n \\ \boldsymbol{w}_m \end{bmatrix} \right\}.$$

By Lemma C.1, for any nonnegative Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{m_1}$ the condition

$$\boldsymbol{C} + \sum_{i=1}^{m_1} \lambda_i^* \boldsymbol{A}_i + \sum_{i=1}^{m_2} \mu_i^* \boldsymbol{B}_i \succeq \boldsymbol{0}$$

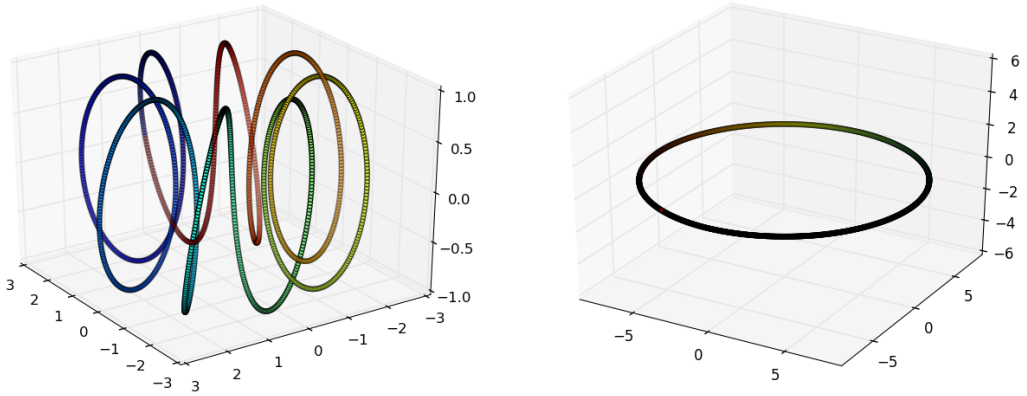
is clearly equivalent to the condition

$$\tilde{\boldsymbol{C}} + \sum_{i=1}^{m_1} \lambda_i^* \tilde{\boldsymbol{A}}_i + \sum_{i=1}^{m_2} \mu_i^* \tilde{\boldsymbol{B}}_i \succeq \boldsymbol{0},$$

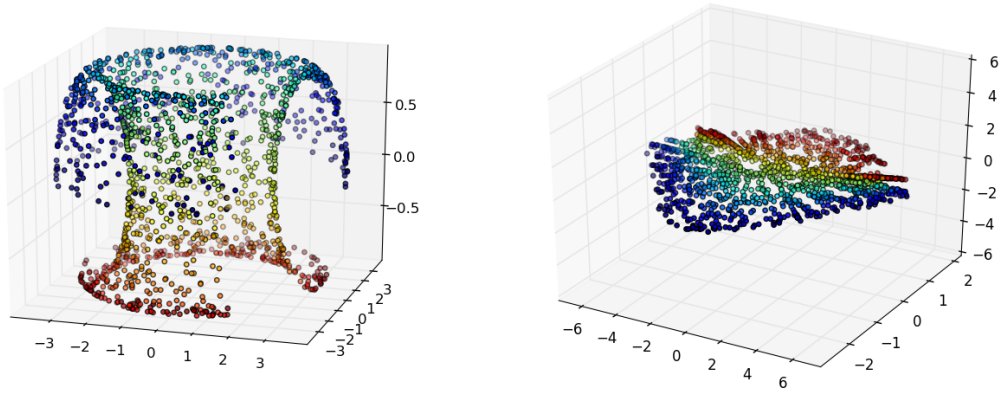
where the matrices $\tilde{\boldsymbol{C}}$, $\tilde{\boldsymbol{A}}_i$ and $\tilde{\boldsymbol{B}}_i$ are defined according to (36)–(38).

Furthermore, optimal solutions to the inequality-constrained problems (39) and (41) have nonnegative Lagrange multipliers corresponding to the inequality constraints. This follows from the definition of KKT conditions for inequality-constrained problems (see e.g. [4]). It is also clear from the KKT conditions of problems (40) and (42) that their optimal solutions always have nonnegative Lagrange multipliers corresponding to the constraints involving the matrices $\tilde{\boldsymbol{A}}_i$.

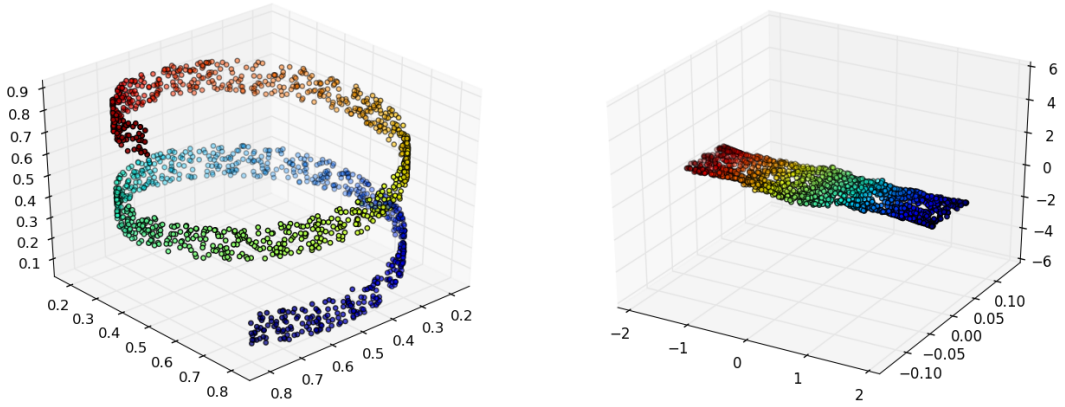
D The Test Datasets and the MVU Results



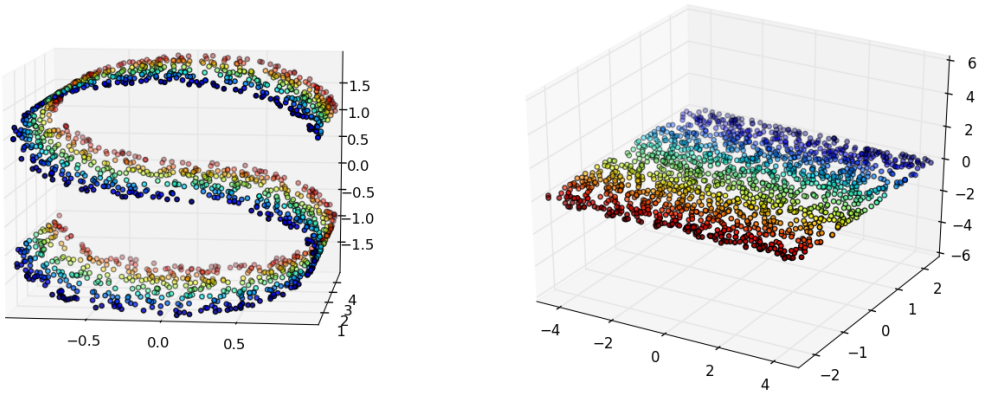
(a) Helix



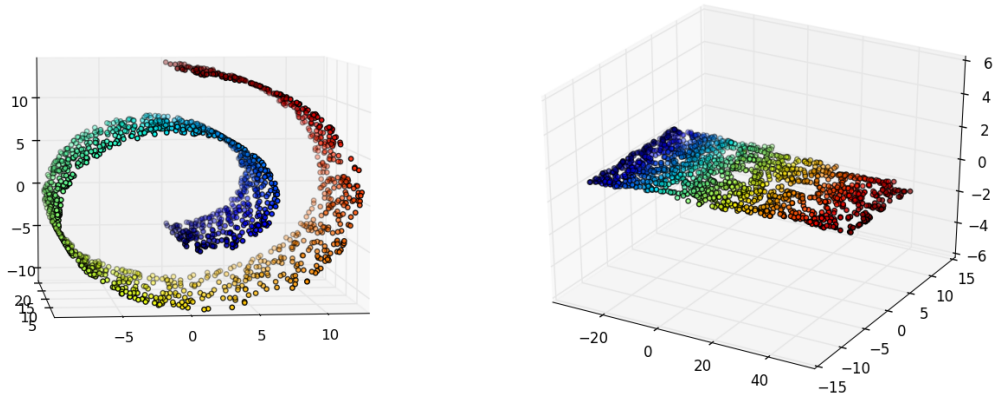
(b) Incomplete tire



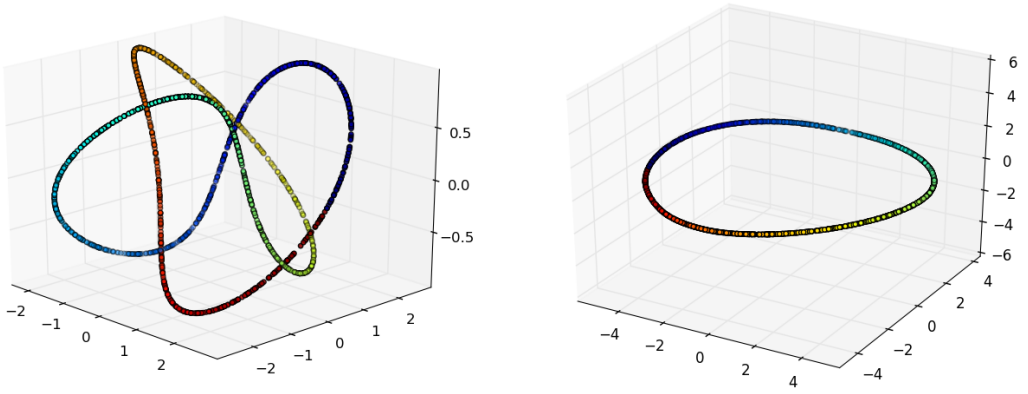
(c) Spiral



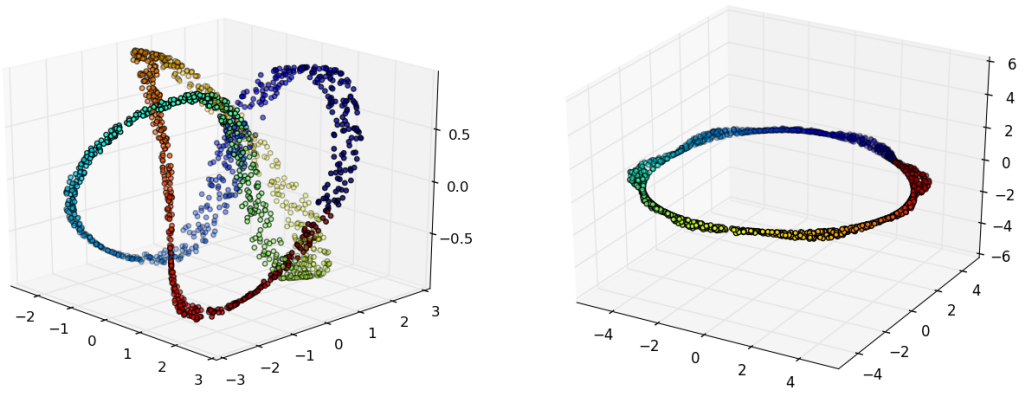
(a) S-roll



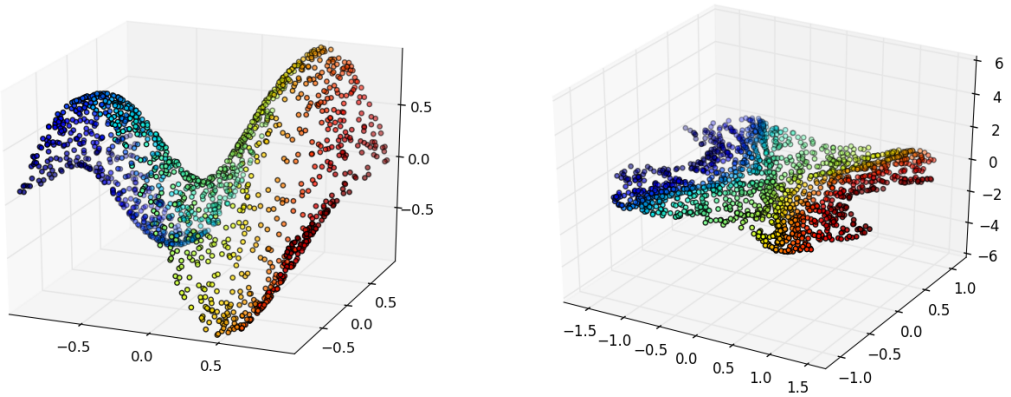
(b) Swiss roll



(c) Trefoil knot



(a) Trefoil ribbon



(b) Twin peaks

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 978-952-12-2859-9
ISSN 1239-1891