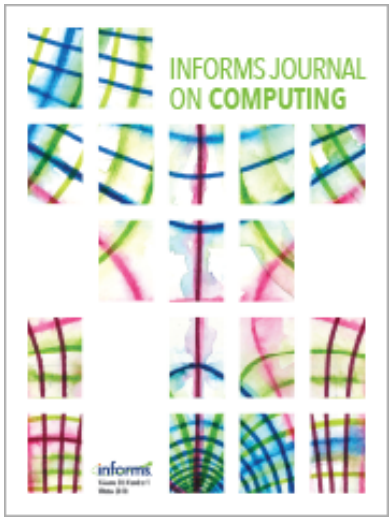


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Peaceman-Rachford Splitting Method for the Protein Side-Chain Positioning Problem

Forbes Burkowski, Haesol Im, Henry Wolkowicz

To cite this article:
Forbes Burkowski, Haesol Im, Henry Wolkowicz (2024) A Peaceman-Rachford Splitting Method for the Protein Side-Chain Positioning Problem. INFORMS Journal on Computing

Published online in Articles in Advance 04 Oct 2024

. https://doi.org/10.1287/ijoc.2023.0094

https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–15
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

A Peaceman-Rachford Splitting Method for the Protein Side-Chain Positioning Problem

Forbes Burkowski,[a] Haesol Im,[b,*] Henry Wolkowicz[b]

[a] Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada; [b] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

*Corresponding author

Contact: fjburkowski@uwaterloo.ca (FB); j5im@uwaterloo.ca, https://orcid.org/0000-0002-2625-9200 (HI); hwolkowi@uwaterloo.ca (HW)

Received: March 24, 2023
Revised: April 23, 2023; December 27, 2023; June 27, 2024; July 16, 2024
Accepted: August 9, 2024
Published Online in Articles in Advance: October 4, 2024

https://doi.org/10.1287/ijoc.2023.0094

Abstract. This paper considers the NP-hard *protein side-chain positioning (SCP)* problem, an important final task of protein structure prediction. We formulate the SCP as an integer quadratic program and derive its doubly nonnegative (DNN) (convex) relaxation. Strict feasibility fails for this DNN relaxation. We apply facial reduction to regularize the problem. This gives rise to a natural splitting of the variables. We then use a variation of the *Peaceman-Rachford splitting method* to solve the DNN relaxation. The resulting relaxation and rounding procedures provide strong approximate solutions. Empirical evidence shows that *almost all* our instances of this NP-hard SCP problem, taken from the Protein Data Bank, are *solved to provable optimality*. Our large problems correspond to solving a DNN relaxation with 2,883,601 binary variables to provable optimality.

History: Accepted by Paul Brooks, Area Editor for Applications in Biology, Medicine, & Healthcare.
Funding: This research was supported by the Natural Sciences and Engineering Research Council of Canada [Grants 50503-10827 and RGPIN-2016-04660].
Supplemental Material: The software that supports the findings of this study is available within the paper and its Supplemental Information (https://pubsonline.informs.org/doi/suppl/10.1287/ijoc.2023.0094) as well as from the IJOC GitHub software repository (https://github.com/INFORMSJoC/2023.0094). The complete IJOC Software and Data Repository is available at https://informsjoc.github.io/.

Keywords: protein structure prediction • side-chain positioning • doubly nonnegative relaxation • facial reduction • Peaceman-Rachford splitting method

1. Introduction

In this paper, we consider the NP-hard *protein side-chain positioning (SCP)* problem, an important final task of protein structure prediction. We formulate the SCP as an integer quadratic program (IQP) and derive its doubly nonnegative (DNN) (convex) relaxation. Strict feasibility fails for this DNN relaxation. We apply facial reduction (FR) to regularize the problem. This gives rise to a natural splitting of the variables. We then use a variation of the *Peaceman-Rachford splitting method (PRSM)* to solve the DNN relaxation. The resulting relaxation and rounding procedures provide strong approximate solutions. Surprisingly, empirical evidence shows that *almost all* our instances of this NP-hard SCP problem, taken from the Protein Data Bank (PDB), are *solved to provable optimality*. Our large problem solutions correspond to solving a DNN relaxation with 2,883,601 binary variables to provable optimality, see Section A and Remark A.1 in the Online Appendix.

The applications of SCP extend to ligand binding (Laudet and Gronemeyer 2002, Looger et al. 2003) and protein-protein docking with backbone flexibility (Wang et al. 2007, Marze et al. 2018). A protein is a macromolecule consisting of a long main chain backbone that provides a set of anchors for a sequence of amino acid side-chains. The backbone is comprised of a repeating triplet of atoms (nitrogen, carbon, carbon) with the central carbon atom being designated as the alpha carbon. An amino acid side-chain is a smaller (1–18 atoms) side branch that is anchored to an alpha carbon. The positions of the atoms in a side-chain can be established by knowing the three-dimensional (3D) position of its alpha carbon and the dihedral angles defined by atoms in the side-chain. The number of dihedral angles varies from one to five depending on the length of the side-chain. This is true for 18 of the 20 amino acids, with glycine and alanine being exceptions because their low atom counts preclude dihedral angles.

It has been observed that the values of dihedral angles are not uniformly distributed. They tend to form clusters with cluster centers that are equally separated (with approximate values, (+60, 180, −60) depending on the

length of the side chain). Consequently, if the dihedral angles are unknown, we at least have a reasonable estimate of their values by appealing to these discretized values. With this strategy being applied, a side-chain with one dihedral angle would have three possible sets of positions for its atoms. We refer to each set of atomic positions as a rotamer. A side-chain with two dihedral angles will have three times three or nine different arrangements of the atoms (i.e., nine rotamers). Three dihedral angles will result in 27 rotamers and four dihedral angles will give 81 rotamers.

In the SCP problem, we are given the coordinates of all atoms in the fixed backbone along with a designation of the amino acid type attached to each alpha carbon. Knowing the coordinates of the backbone atoms allows us to compute the backbone dihedral angles associated with a particular alpha carbon atom. These two angles along with the amino acid type provide the information that is needed to build a rotamer database query. We have used the rotamer database provided by the Dunbrack Laboratory (Dunbrack and Karplus 1993). The response to such a query provides a set of rotamers most suitable for the amino acid at that alpha carbon position (and consistent with the specified backbone dihedral angles). As noted earlier, the number of rotamers for a particular amino acid position will be 3, 9, 27, or 81 depending on the amino acid type. To solve the SCP problem, we need to select a particular residue from each rotamer set so that the entire collection of selected residues yields the lowest energy for the system. In particular, we must avoid the selection of a rotamer that collides with the selected rotamer of any neighboring side chain. After the SCP algorithm is completed, we will have a set of optimal rotamers. Because each of the selected rotamers is a member of the rotamer library, it has a discretized set of χ_1 and χ_2 angles and should be regarded as a reasonable approximation of the true conformation that would be seen in the actual molecule (Xu and Berger 2006, Burkowski 2015). In practice, a biochemist would execute the SCP algorithm and follow this analysis with a molecular dynamics simulation (or perhaps a simple energy minimization procedure) that would then refine the chi angles to get values that are much closer to those observed in the molecule. Output from the SCP algorithm will provide the initial conformations necessary for the molecular dynamics program. As an aside, it should be noted that the utilization of a molecular dynamics program with arbitrary (or random) initial side-chain angle settings will not be successful in generating the full set of chi angles because the system will simply settle into a local energy minimum that is typically much higher than the empirically observed energy minimum. See Section 4.2 for more details on constructing the energies between the rotamers.

The SCP problem is known to be NP-hard (Akutsu 1997). The nature of the SCP problem has motivated the development of many heuristic based algorithms (Desmet et al. 1992, Bower et al. 1997, Samudrala and Moult 1998, Canutescu et al. 2003, Bahadur et al. 2004, Xu and Berger 2006). Many of these approaches rely on the graph structure of the underlying SCP problem with the rotamers considered as the nodes of the graph. Other approaches for SCP range from probabilistic (Holm and Sander 1991, Lee 1994, Shenkin et al. 1996), integer programming (Eriksson et al. 2001, Althaus et al. 2002, Kingsford et al. 2005), and semidefinite programming (Chazelle et al. 2004, Burkowski et al. 2014). Given a rotamer library, the SCP problem can be modelled using an IQP. We then obtain a *semidefinite programming (SDP)* relaxation to the IQP using a lifting of variables. Strict feasibility fails for the SDP relaxation. FR is then used to regularize the problem. Finally, a DNN relaxation results from adding nonnegativity, as well as other polyhedral constraints.

The FR yields a natural splitting of variables into cone constrained and polyhedral constrained variables. The elegant splitting of variables fits into the framework of splitting methods that allow for simplified subproblems that deal with the split variables individually. The framework gives an efficient procedure of engaging constraints that are difficult to process simultaneously (Oliveira et al. 2018, Li et al. 2019, Graham et al. 2020). We solve the DNN relaxation using a variation of the so-called *Peaceman-Rachford splitting method (PRSM)*. The usage of the splitting method for the DNN relaxation allows for an effective treatment for handling implicit redundant constraints and the ill-posed data that stems from collisions between rotamers.

1.1. Notation

We let $\mathbb{R}^n, \mathbb{R}^{m\times n}$ denote the standard real Euclidean spaces; $\mathbb{S}^n$ denotes the Euclidean space of n-by-n real symmetric matrices; $\mathbb{S}^n_+$ ($\mathbb{S}^n_{++}$, respectively) denotes the cone of n-by-n positive semidefinite (definite, respectively) matrices. We write $\boldsymbol{X} \succeq 0$ if $\boldsymbol{X} \in \mathbb{S}^n_+$, and $\boldsymbol{X} \succ 0$ if $\boldsymbol{X} \in \mathbb{S}^n_{++}$. range($\boldsymbol{X}$) and null($\boldsymbol{X}$) denote the range and null space of $\boldsymbol{X}$, respectively. Given $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, trace($\boldsymbol{X}$) denotes the *trace* of $\boldsymbol{X}$. For two matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{m\times n}$, $\langle \boldsymbol{X}, \boldsymbol{Y}\rangle = \text{trace}(\boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}})$ denotes the usual trace inner product between $\boldsymbol{X}$ and $\boldsymbol{Y}$; $\boldsymbol{X} \circ \boldsymbol{Y}$ denotes the element-wise, or Hadamard, product of $\boldsymbol{X}$ and $\boldsymbol{Y}$. Given a closed convex set $\mathcal{C}$ in a Euclidean space, $\mathcal{N}_{\mathcal{C}}(\boldsymbol{x})$ denotes the *normal cone* of $\mathcal{C}$ at $\boldsymbol{x}$. Given $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, diag($\boldsymbol{X}$) denotes the vector formed from the diagonal entries of $\boldsymbol{X}$. Then, $\text{Diag}(\boldsymbol{v}) = \text{diag}^*(\boldsymbol{v})$ is the adjoint linear transformation that forms the diagonal matrix from the vector $\boldsymbol{v}$. Given a collection of matrices $\{\boldsymbol{A}_i\}_{i=1}^m$, we let $\text{BlkDiag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m)$ denote the block diagonal matrix with the ith diagonal block $\boldsymbol{A}_i$; $\overline{\boldsymbol{e}}_n$ denotes the

n-dimensional ones vector. We omit the subscript when the dimension is clear. Given a positive integer m, $[m]$ is the set $\{1,\dots,m\}$.

1.2. Contributions and Outline

The model for SCP as an IQP is formulated in Section 2. This includes the derivations of the SDP and DNN relaxations. The derivation for the SDP relaxation was first presented in (Burkowski et al. 2014) via Lagrangian relaxation. Here, we present a much simpler derivation via a direct lifting of the variables to symmetric matrix space. Included are the details for the FR that gives rise to a natural splitting of variables and the so-called *gangster constraints*. The PRSM algorithm for the DNN relaxation is presented in Section 3. This includes upper and lower bounding techniques that allow for increased efficiency and accuracy. In Section 4, we use the real-world data from the PDB[1] to illustrate the strength of our approach. The splitting method applied to the DNN relaxation effectively handles collisions between rotamers indicated by large values in the data. Moreover, the numerical experiments demonstrate that our approach *provably* solves *almost all* instances[2] to the global optimum of the NP-hard protein SCP problem.

2. Model Derivation

The goal of this section is to obtain the DNN relaxation of the SCP problem. We start by presenting a formulation of the SCP as an IQP in Section 2.1. We then derive its SDP relaxation in Section 2.2. We continue the derivation by identifying redundant constraints in the IQP and in the SDP relaxation in order to obtain a complete (stable) DNN relaxation in Section 2.3.

2.1. Problem Formulation as IQP

We are given a collection of p disjoint sets of positive integers

$$\mathcal{V}_i := \{v_i^1, v_i^2, \dots, v_i^{m_i}\},\ i = 1,\dots,p.$$

We call each set $\mathcal{V}_i$ a *rotamer set* and its members *rotamers*. We use $n_0 = \sum_{i=1}^p m_i$ and $\mathcal{V} = \cup_{i=1}^p \mathcal{V}_i$. The *protein side-chain positioning problem* seeks to

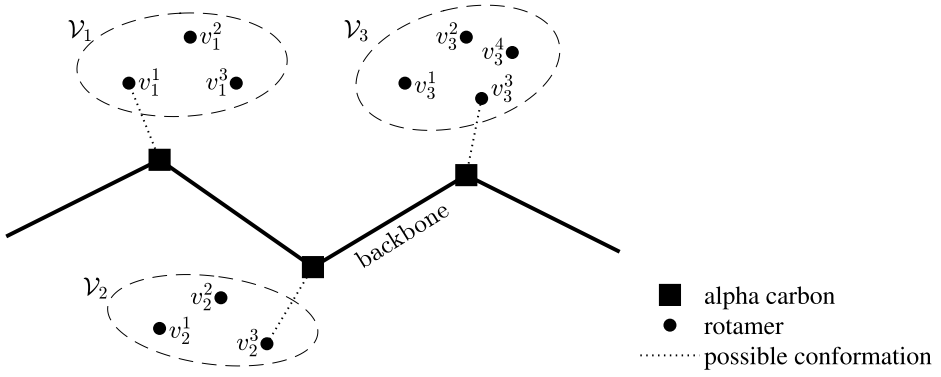
1. Select *exactly one* rotamer v_i^j, from each set $\mathcal{V}_i$, where $j \in [m_i]$ (Figure 1[3]); and
2. Minimize the sum of the weights (energy) between the chosen rotamers plus between each chosen rotamer and the backbone.

By viewing the rotamers as a set of nodes of a graph, the SCP problem can be realized as a discrete optimization problem. We start by setting an energy matrix $\boldsymbol{E} \in \mathbb{S}^{n_0}$ that obeys the following rules:

$$\boldsymbol{E}_{uv} = \begin{cases} \infty & \text{if } u,v \in \mathcal{V}_i \text{ for some } i \in [p], \\ \text{(energy between rotamer } u \text{ and backbone)} & \text{if } u = v, \\ \text{(energy between rotamers } u,v) & \text{otherwise.} \end{cases}$$

UCSF Chimera application[4] is used to obtain a particular setting of the energy matrix for the numerical experiments in Section 4. This records the energy values of pairs of rotamers and between rotamers to the backbone. We use the convention that $0 \cdot \infty = 0$ when adding up the weights (energies). This is simplified in our algorithm by using constraints so that exactly one rotamer from each set is chosen. Moreover, each diagonal block of E of

Figure 1. Diagram of the Protein Side-Chain Positioning Problem



size m_i can be assumed to be a diagonal matrix as we are looking to choose exactly one rotamer per set $\mathcal{V}_i$ (see Section 4.2 for more details on constructing the energies between the rotamers).

The resulting IQP over the indicator vector $\boldsymbol{x}$ is

$$\begin{aligned} p^*_{\text{IQP}} := \min_{\boldsymbol{x}} \quad & \sum_{u,v} \boldsymbol{E}_{uv}\boldsymbol{x}_u\boldsymbol{x}_v \\ \text{s.t.} \quad & \sum_{u\in\mathcal{V}_k} \boldsymbol{x}_u = 1,\ k = 1,\dots,p \\ & \boldsymbol{x}_u \in \{0,1\},\ \ \forall u \in \mathcal{V}. \end{aligned} \tag{2.1}$$

The constraints in (2.1) ensures that exactly one rotamer is chosen for each rotamer set $\mathcal{V}_i$. Denote the block diagonal matrix

$$\boldsymbol{A} = \text{BlkDiag}(\overline{\boldsymbol{e}}^T_{m_1}, \overline{\boldsymbol{e}}^T_{m_2}, \dots, \overline{\boldsymbol{e}}^T_{m_p}) \in \mathbb{R}^{p\times n_0}. \tag{2.2}$$

Then $\boldsymbol{A}\boldsymbol{x} = \overline{\boldsymbol{e}}_p$ is a concise representation of the first equality constraints in (2.1). This yields the following representation of the SCP problem:

$$\begin{aligned} (IQP) \qquad p^*_{\text{IQP}} = \min_{\boldsymbol{x}} \ & \boldsymbol{x}^T\boldsymbol{E}\boldsymbol{x} \\ \text{s.t.} \ & \boldsymbol{A}\boldsymbol{x} = \overline{\boldsymbol{e}}_p \\ & \boldsymbol{x} \in \{0,1\}^{n_0}. \end{aligned} \tag{2.3}$$

2.2. SDP Relaxation

The SCP (2.3) is NP-hard (Akutsu 1997). In the remainder of this section, we discuss (convex) relaxations for the problem. In particular, we derive a DNN to (2.3). An equivalent formulation of the DNN is proposed in Burkowski et al. (2014). The derivation in Burkowski et al. (2014) begins by replacing the linear constraint in (2.3) by $\|\boldsymbol{A}\boldsymbol{x} - \overline{\boldsymbol{e}}_p\|^2 = 0$ and the binary constraints by $\boldsymbol{x}\circ\boldsymbol{x} - \boldsymbol{x} = 0$. Then the Lagrangian relaxation to the quadratically constrained model results in a dual program, L_D, that lower bounds p^*_{IQP} and satisfies the Slater constraint qualification. Finally, a dual to the dual L_D is formulated to obtain a relaxation to (2.3). In this paper, we present a simplified derivation for the SDP and DNN relaxations to (2.3) via a simple direct lifting.

Let

$$\hat{\boldsymbol{E}} := \text{BlkDiag}(0, \boldsymbol{E}) \in \mathbb{S}^{n_0+1}, \quad \boldsymbol{E}_{00} := \boldsymbol{e}_0\boldsymbol{e}_0^T \in \mathbb{S}^{n_0+1},$$

where $\boldsymbol{e}_0$ is the first unit vector. In this section, we aim to obtain the following SDP relaxation to the discrete optimization problem (2.3):

$$\begin{aligned} (SDP) \qquad p^*_{\text{SDP}} := \min_{\boldsymbol{R},\boldsymbol{Y}} \quad & \text{trace}(\hat{\boldsymbol{E}}\boldsymbol{Y}) \\ & G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = \boldsymbol{E}_{00} \\ & \boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^{\boldsymbol{T}} \\ & \boldsymbol{R} \in \mathbb{S}^{n_0+1-p}_+, \end{aligned} \tag{2.4}$$

where the so-called *gangster constraint* $G_{\hat{\mathcal{J}}}(\cdot)$ is described in Section 2.2.1 and the matrices $\boldsymbol{V}, \boldsymbol{R}$ are defined in Section 2.2.2.

The first step for deriving the SDP relaxation (2.4) is to *lift* the variables. Given $\boldsymbol{x} \in \mathbb{R}^{n_0}$, we lift to symmetric matrix space using the rank-one *lifted matrix*

$$\boldsymbol{Y}_{\boldsymbol{x}} := \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}\begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}^T = \begin{bmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{x}\boldsymbol{x}^T \end{bmatrix} \in \mathbb{S}^{n_0+1}.$$

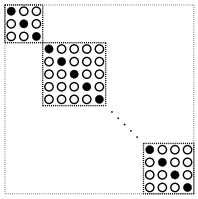
For the SDP relaxation, we index the rows and columns *starting from zero*, that is, the row and column indices are $\{0,1,\dots,n_0\}$. This lifting allows for an alternative representation of the objective function

$$\boldsymbol{x}^T\boldsymbol{E}\boldsymbol{x} = \left\langle \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{E} \end{bmatrix}, \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}\begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}^T \right\rangle = \langle \hat{\boldsymbol{E}}, \boldsymbol{Y}_x \rangle.$$

The convex relaxation is obtained by relaxing the hard nonconvex constraint of maintaining $\boldsymbol{Y}$ rank one (see below). We now show how this lifting process gives rise to the constraints of Model (2.4):

1. The linear (gangster) constraint $G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = \boldsymbol{E}_{00}$ (Section 2.2.1);
2. $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T$, where $\boldsymbol{R} \in \mathbb{S}^{n_0+1-p}_+$ (Section 2.2.2).

Figure 2. Index Set $\mathcal{J}$ of Zeros



Note. Members of $\mathcal{J}$ correspond to the off-diagonal elements of diagonal blocks of W indicated by the symbol ○.

2.2.1. Gangster Constraint $G_{\hat{\mathcal{J}}}(Y) = E_{00}$.

Let $W \in \mathbb{S}^{n_0}$ be given, and define the set of indices

$$\mathcal{J} := \left\{ \left(\sum_{i=1}^{j} m_{i-1} + k, \sum_{i=1}^{j} m_{i-1} + \ell \right) : j \in \{1, \dots, p-1\}, k, \ell \in \{2, \dots, m_i - 1\}, k \neq \ell \right\}.$$

Here, m_i is the cardinality of rotamer set $\mathcal{V}_i$, and $m_0 = 0$. In other words, $\mathcal{J}$ is the set of off-diagonal indices of the m_i-by-m_i diagonal blocks of $\boldsymbol{W} \in \mathbb{S}^{n_0}$; see Figure 2 for a visual illustration of the positioning of these indices. Note that these indices correspond to exactly

$$\boldsymbol{W}_{uv} = x_u x_v = 0, u \neq v, u, v \in \mathcal{V}_i,$$

that is, two distinct rotamers in the same rotamer set cannot be chosen.

With the above set of indices, we define the mapping

$$G_{\mathcal{J}} : \mathbb{S}^{n_0} \to \mathbb{R}^{|\mathcal{J}|} \text{ by } G_{\mathcal{J}}(\boldsymbol{W}) = (\boldsymbol{W}_{ij})_{ij \in \mathcal{J}}.$$

Alternatively, we also view the mapping $G_{\mathcal{J}}$ as the operator from $\mathbb{S}^{n_0}$ to $\mathbb{S}^{n_0}$:

$$G_{\mathcal{J}} : \mathbb{S}^{n_0} \to \mathbb{S}^{n_0}, \ (G_{\mathcal{J}}(\boldsymbol{W}))_{i,j} = \begin{cases} \boldsymbol{W}_{i,j} & \text{if } (i,j) \text{ or } (j,i) \in \mathcal{J}, \\ 0 & \text{otherwise.} \end{cases}$$

The map $G_{\mathcal{J}}$ can also be viewed as the operator on $\mathbb{S}^{n_0}$ defined by $G_{\mathcal{J}}(\boldsymbol{W}) = (\boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{I}) \circ \boldsymbol{W}$ with $\boldsymbol{A}$ defined in (2.2). Recall $\circ$ is the element-wise matrix product. In other words, $G_{\mathcal{J}}(\boldsymbol{W})$ is the projection that chooses elements of $\boldsymbol{W}$ corresponding to the index set $\mathcal{J}$. The constraint $G_{\mathcal{J}}(\boldsymbol{W}) = 0$ is often called the *gangster constraint* as it fixes elements of $\boldsymbol{W}$ with indices in $\mathcal{J}$ to zero (shoots holes in the matrix). The index set and operator are now extended to lifted variables in $\mathbb{S}^{n_0+1}$ with

$$\hat{\mathcal{J}} := \{(0,0)\} \cup \mathcal{J} \subset \{0, 1, \dots, n_0\} \times \{0, 1, \dots, n_0\}$$

and

$$G_{\hat{\mathcal{J}}} : \mathbb{S}^{n_0+1} \to \mathbb{R}^{|\hat{\mathcal{J}}|}, \ G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = (\boldsymbol{Y}_{ij})_{ij \in \hat{\mathcal{J}}}.$$

This yields the *gangster constraint* as a projection and as an operator, respectively,

$$G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = \boldsymbol{e}_0 \in \mathbb{R}^{1+|\mathcal{J}|}, \quad G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = E_{00}.$$

2.2.2. Facial Reduction.

We now derive the constraint $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T$, $\boldsymbol{R} \in \mathbb{S}_+^{n_0+1-p}$. Let $\boldsymbol{x}$ be a feasible solution to (2.3) and observe that

$$\begin{aligned} \boldsymbol{A}\boldsymbol{x} = \overline{\boldsymbol{e}}_p &\Rightarrow \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}^T \begin{bmatrix} -\overline{\boldsymbol{e}}_p^T \\ \boldsymbol{A}^T \end{bmatrix} = 0 \\ &\Rightarrow \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}^T \begin{bmatrix} -\overline{\boldsymbol{e}}_p^T \\ \boldsymbol{A}^T \end{bmatrix} \begin{bmatrix} -\overline{\boldsymbol{e}}_p^T \\ \boldsymbol{A}^T \end{bmatrix}^T = 0 \\ &\Rightarrow \left\langle \underbrace{\begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}^T}_{=\boldsymbol{Y}_{\boldsymbol{x}}}, \underbrace{\begin{bmatrix} -\overline{\boldsymbol{e}}_p^T \\ \boldsymbol{A}^T \end{bmatrix} \begin{bmatrix} -\overline{\boldsymbol{e}}_p^T \\ \boldsymbol{A}^T \end{bmatrix}^T}_{=:\boldsymbol{K}} \right\rangle = 0. \end{aligned}$$

Because both arguments in the last inner product are positive semidefinite, we obtain the useful property:

$$\langle \boldsymbol{K}, \boldsymbol{Y}_x \rangle = 0 \Rightarrow \boldsymbol{K}\boldsymbol{Y}_x = 0 \Rightarrow \text{range}(\boldsymbol{Y}_x) \subseteq \text{null}(\boldsymbol{K}). \tag{2.5}$$

In other words, null($\boldsymbol{K}$) yields the range that all lifted feasible points can have.

Now let $\boldsymbol{V} \in \mathbb{R}^{(n_0+1)\times(n_0+1-p)}$ be a full-column rank matrix such that

$$\text{range}(\boldsymbol{V}) = \text{null}(\boldsymbol{K}) = \text{null}\left(\begin{bmatrix} -\overline{\boldsymbol{e}}_p^T \\ \boldsymbol{A}^T \end{bmatrix}^T \right). \tag{2.6}$$

For our purposes, we choose $\boldsymbol{V}$ with normalized columns. Because $\boldsymbol{A}$ is full-row rank, we get that rank($\boldsymbol{K}$) = p. Finally, we can represent any feasible $\boldsymbol{Y}_x$ using $\boldsymbol{V}$:

$$\boldsymbol{Y}_x \in \boldsymbol{V}\mathbb{S}_+^{n_0+1-p}\boldsymbol{V}^T.$$

This is the well-known *facial reduction* technique (Drusvyatskiy and Wolkowicz 2017). The matrix $\boldsymbol{K}$ is an *exposing vector* for the feasible set. The matrix $\boldsymbol{V}$ is known as a *facial range vector*.

The remaining step for the SDP relaxation is simple. We note that we require rank($\boldsymbol{Y}_x$) = 1, a nonconvex constraint. We discard this hard rank restriction and obtain a convex relaxation variables of the form

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T \text{ where } \boldsymbol{R} \in \mathbb{S}_+^{n_0+1-p}.$$

This completes the derivation of the relaxation in (2.4). It is known that there is a $\hat{\boldsymbol{R}} \in \mathbb{S}_{++}^{n_0+1-p}$ (strictly) feasible for (2.4); see Burkowski et al. (2014).

2.3. DNN Relaxation

We continue with the SDP relaxation derived in Section 2.2 to complete our relaxation by adding additional constraints to (2.4). In Theorem 2.1, we obtain two additional properties of Model (2.4).

Theorem 2.1. *Suppose that ($\boldsymbol{R}$, $\boldsymbol{Y}$) are feasible to (2.4). Then the following hold.*

1. *The first column of $\boldsymbol{Y}$ is equal to the diagonal of $\boldsymbol{Y}$.*
2. trace($\boldsymbol{R}$) = $1 + p$.

Proof. We recall that range($\boldsymbol{V}$) = null($[\, -\overline{\boldsymbol{e}}_p \quad \boldsymbol{A}\,]$) from (2.6). Hence, we have

$$[\, -\overline{\boldsymbol{e}}_p \quad \boldsymbol{A}\,]\boldsymbol{Y} = [\, -\overline{\boldsymbol{e}}_p \quad \boldsymbol{A}\,]\boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T = \boldsymbol{0}\boldsymbol{R}\boldsymbol{V}^{\mathrm{T}} = 0 \tag{2.7}$$

and exploit the structure of $[\, -\overline{\boldsymbol{e}}_p \quad \boldsymbol{A}\,]\boldsymbol{Y}$. We first partition $\boldsymbol{Y}$ as follows:

$$\boldsymbol{Y} = \begin{bmatrix} 1 & \boldsymbol{Y}_{10}^T & \boldsymbol{Y}_{20}^T & \cdots & \boldsymbol{Y}_{p0}^T \\ \boldsymbol{Y}_{10} & \boldsymbol{Y}_{11} & \boldsymbol{Y}_{12} & \cdots & \boldsymbol{Y}_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{Y}_{p0} & \boldsymbol{Y}_{p1} & \boldsymbol{Y}_{p2} & \cdots & \boldsymbol{Y}_{pp} \end{bmatrix} \in \mathbb{S}^{n_0+1}, \tag{2.8}$$

where $\boldsymbol{Y}_{ii} \in \mathbb{S}^{m_i}$, $\boldsymbol{Y}_{ij} \in \mathbb{R}^{m_i \times m_j}$, $\boldsymbol{Y}_{i0} \in \mathbb{R}^{m_i}$, $\forall i, j \in [p]$. Let $\boldsymbol{Y}_{ij}^{\mathbf{col}\,\ell}$ denote the ℓth column of the (i, j)th block of $\boldsymbol{Y}$ and $\boldsymbol{Y}_{i0,\ell}$ to denote the ℓth coordinate of the vector $\boldsymbol{Y}_{i0} \in \mathbb{R}^{m_i}$.

Then expanding $[\, -\overline{\boldsymbol{e}}_p \quad \boldsymbol{A}\,]\boldsymbol{Y}$ with the block representation (2.8) yields

$$[\, -\overline{\boldsymbol{e}}_p \quad \boldsymbol{A}\,]\boldsymbol{Y} = [\, \boldsymbol{a}_0 \quad \boldsymbol{A}_1 \quad \cdots \quad \boldsymbol{A}_p\,] \in \mathbb{R}^{p\times(n_0+1)},$$

where

$$\boldsymbol{a}_0 = \begin{bmatrix} -1 + \overline{\boldsymbol{e}}_{m_1}^T \boldsymbol{Y}_{10} \\ -1 + \overline{\boldsymbol{e}}_{m_2}^T \boldsymbol{Y}_{20} \\ \vdots \\ -1 + \overline{\boldsymbol{e}}_{m_p}^T \boldsymbol{Y}_{p0} \end{bmatrix} \in \mathbb{R}^p. \tag{2.9}$$

Also, for each $i \in [p]$,

$$\boldsymbol{A}_i = \begin{bmatrix} -\boldsymbol{Y}_{i0,1} + \overline{\boldsymbol{e}}_{m_1}^T \boldsymbol{Y}_{1i}^{\mathbf{col}\,1} & -\boldsymbol{Y}_{i0,2} + \overline{\boldsymbol{e}}_{m_1}^T \boldsymbol{Y}_{1i}^{\mathbf{col}\,2} & \cdots & -\boldsymbol{Y}_{i0,m_i} + \overline{\boldsymbol{e}}_{m_1}^T \boldsymbol{Y}_{1i}^{\mathbf{col}\,m_i} \\ \vdots & \vdots & \ddots & \vdots \\ -\boldsymbol{Y}_{i0,1} + \overline{\boldsymbol{e}}_{m_j}^T \boldsymbol{Y}_{ji}^{\mathbf{col}\,1} & -\boldsymbol{Y}_{i0,2} + \overline{\boldsymbol{e}}_{m_j}^T \boldsymbol{Y}_{ji}^{\mathbf{col}\,2} & \cdots & -\boldsymbol{Y}_{i0,m_i} + \overline{\boldsymbol{e}}_{m_j}^T \boldsymbol{Y}_{ji}^{\mathbf{col}\,m_i} \\ \vdots & \vdots & \ddots & \vdots \\ -\boldsymbol{Y}_{i0,1} + \overline{\boldsymbol{e}}_{m_p}^T \boldsymbol{Y}_{pi}^{\mathbf{col}\,1} & -\boldsymbol{Y}_{i0,2} + \overline{\boldsymbol{e}}_{m_p}^T \boldsymbol{Y}_{pi}^{\mathbf{col}\,2} & \cdots & -\boldsymbol{Y}_{i0,m_i} + \overline{\boldsymbol{e}}_{m_p}^T \boldsymbol{Y}_{pi}^{\mathbf{col}\,m_i} \end{bmatrix} \in \mathbb{R}^{p \times m_i}.$$

By (2.7), we have $\boldsymbol{A}_i = 0, \ \forall i \in [p]$. Thus, for each $i \in [p]$, the ith row of $\boldsymbol{A}_i$ yields

$$\boldsymbol{Y}_{i0,\ell} = \overline{\boldsymbol{e}}_{m_i}^T \boldsymbol{Y}_{ii}^{\mathbf{col}\,\ell}, \ \ell \in [m_i].$$

Because $G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = \boldsymbol{E}_{00}$ holds, we see that

$$\operatorname{diag}(\boldsymbol{Y}_{ii}) = \boldsymbol{Y}_{i0}, \ \ \forall i \in [p].$$

Consequently, the first column and the diagonal of $\boldsymbol{Y}$ are identical.

We now show that $\operatorname{trace}(\boldsymbol{R}) = 1 + p$. By (2.7), the vector $\boldsymbol{a}_0$ from (2.9) is zero. Thus, we obtain

$$\overline{\boldsymbol{e}}_{m_i}^T \boldsymbol{Y}_{i0} = 1, \ \ \forall i \in [p].$$

Because $\operatorname{diag}(\boldsymbol{Y}_{ii}) = \boldsymbol{Y}_{i0}, \ \ \forall i \in [p]$ from Item 1, we must have that $\operatorname{trace}(\boldsymbol{Y}_{\mathrm{ii}}) = 1, \ \ \forall \mathrm{i} \in [\mathrm{p}]$. Hence, $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T$ gives

$$1 + p = \operatorname{trace}(\boldsymbol{Y}) = \operatorname{trace}(\boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T) = \operatorname{trace}(\boldsymbol{R}),$$

where the last equality holds because $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$. □

Item 1 of Theorem 2.1 is known in the literature as the *arrow* constraint and arises from the Lagrangian dual (Burkowski et al. 2014). The derivation herein exploits the steps from the direct lifting.

We recall that the original model (2.3) has binary constraints on the variables $\boldsymbol{x}$, and the direct lifting yields a rank one variable of the form $\begin{bmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{x}\boldsymbol{x}^T \end{bmatrix} \in \mathbb{S}^{n_0+1}$. Hence, we can strengthen our model by including the constraint $\boldsymbol{Y}_{i,j} \in [0,1], \ \forall i,j$.

We define the sets

$$\mathcal{R} := \{\boldsymbol{R} \in \mathbb{S}^{n_0+1-p} : \boldsymbol{R} \geq 0, \ \operatorname{trace}(\boldsymbol{R}) = p + 1\},$$
$$\mathcal{Y} := \{\boldsymbol{Y} \in \mathbb{S}^{n_0+1} : G_{\hat{\mathcal{J}}}(\boldsymbol{Y}) = \boldsymbol{E}_{00}, \ 0 \leq \boldsymbol{Y} \leq 1\}.$$

By including the additional constraints $\operatorname{trace}(\boldsymbol{R}) = 1 + p$ and $0 \leq \boldsymbol{Y} \leq 1$ to the SDP relaxation (2.4), we obtain the DNN relaxation to (2.3):

$$(DNN) \qquad \begin{aligned} p^*_{\mathrm{DNN}} := \min_{\boldsymbol{R},\boldsymbol{Y}} \ & \operatorname{trace}(\hat{\boldsymbol{E}}\boldsymbol{Y}) \\ & \boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T \\ & \boldsymbol{R} \in \mathcal{R} \\ & \boldsymbol{Y} \in \mathcal{Y}. \end{aligned} \tag{2.10}$$

Note that both (DNN) and (SDP) are relaxations to (IQP), but (DNN) is a stronger model than (SDP), that is,

$$p^*_{\mathrm{SDP}} \leq p^*_{\mathrm{DNN}} \leq p^*_{\mathrm{IQP}}.$$

In addition, there are redundant constraints in Model (2.10). These (implicit) redundant constraints result in numerical instabilities when they are not handled properly. In Section 2.3, we use the splitting method to distribute constraints to two different subproblems wherein they are not redundant but in fact strengthen the subproblems.

We demonstrate the strength of (DNN) in Section 4.3.1. The DNN relaxation has a linear objective with an *onto* linear equality constraint, and compact, convex, feasible set constraints. The first-order optimality conditions

for (2.10) are

$$\begin{aligned} &0 \in -\boldsymbol{V}^T\boldsymbol{Z}\boldsymbol{V} + \mathcal{N}_{\mathcal{R}}(\boldsymbol{R}), && \text{(dual feasibility with respect to } \boldsymbol{R}) \\ &0 \in \hat{\boldsymbol{E}} + \boldsymbol{Z} + \mathcal{N}_{\mathcal{Y}}(\boldsymbol{Y}), && \text{(dual feasibility with respect to } \boldsymbol{Y}) \\ &\boldsymbol{Y} = \hat{\boldsymbol{V}}\boldsymbol{R}\hat{\boldsymbol{V}}^T, \quad \boldsymbol{R} \in \mathcal{R}, \boldsymbol{Y} \in \mathcal{Y}, && \text{(primal feasibility)} \end{aligned} \tag{2.11}$$

where $\mathcal{N}_{\mathcal{R}}(\boldsymbol{R}), \mathcal{N}_{\mathcal{Y}}(\boldsymbol{Y})$ are the normal cones and $\boldsymbol{Z}$ is a Lagrange multiplier associated with the constraint $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T$. Theorem 2.2 states that some elements of the dual optimal multiplier $\boldsymbol{Z}^*$ are known in advance. We take advantage of this fact in our algorithm for solving the DNN in Section 3.

Theorem 2.2. *Let* $(\boldsymbol{R}^*, \boldsymbol{Y}^*)$ *be an optimal pair for* (2.10)*, and let*

$$\mathcal{Z}_A := \{\boldsymbol{Z} \in \mathbb{S}^{n_0+1} : \boldsymbol{Z}_{i,i} = -(\hat{\boldsymbol{E}})_{i,i}, \boldsymbol{Z}_{0,i} = \boldsymbol{Z}_{i,0} = -(\hat{\boldsymbol{E}})_{0,i}, i = 1, \ldots, n_0\}.$$

Then there exists $\boldsymbol{Z}^* \in \mathcal{Z}_A$ *such that* $(\boldsymbol{R}^*, \boldsymbol{Y}^*, \boldsymbol{Z}^*)$ *solves* (2.11).

Proof. The proof uses the optimality conditions (2.11) and Theorem 2.1. The proof can be found in Graham et al. (2020, theorem 2.11). □

3. Algorithm

In this section, we present the algorithm for solving the DNN relaxation (2.10). For $\beta > 0$, define the augmented Lagrangian $\mathcal{L}_A$ of Model (2.10) as

$$\mathcal{L}_A(\boldsymbol{R}, \boldsymbol{Y}, \boldsymbol{Z}) := \langle \hat{\boldsymbol{E}}, \boldsymbol{Y} \rangle + \langle \boldsymbol{Z}, \boldsymbol{Y} - \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T \rangle + \frac{\beta}{2} \|\boldsymbol{Y} - \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T\|_F^2. \tag{3.1}$$

Let $\mathcal{P}_{\mathcal{Z}_0}(\boldsymbol{Z})$ denote the linear projection operator onto the linear manifold

$$\mathcal{Z}_0 = \{\boldsymbol{Z} \in \mathbb{S}^{n_0+1} : \boldsymbol{Z}_{i,i} = \boldsymbol{Z}_{0,i} = \boldsymbol{Z}_{i,0} = 0, i = 1, \ldots, n_0\}.$$

In other words, the projection operator $\mathcal{P}_{\mathcal{Z}_0}(\boldsymbol{Z})$ sets the first column, first row, and the diagonal elements of $\boldsymbol{Z}$ to be zero, except for the (0, 0)th entry.

We use the *restricted dual PRSM (rPRSM)* (Algorithm 1), a variation of the strictly contractive PRSM to solve Model (2.10).

Algorithm 1 (rPRSM (Graham et al. 2020) for Solving (2.10))

1: **Initialize:** $\boldsymbol{Y}^0 \in \mathbb{S}^{n_0+1}$, $\boldsymbol{Z}^0 \in \mathcal{Z}_A$, $\beta \in (0, \infty)$, $\gamma \in (0, 1)$
2: **while** termination criteria are not met **do**
3: $\quad \boldsymbol{R}^{k+1} = \underset{\boldsymbol{R} \in \mathcal{R}}{\arg\min}\ \mathcal{L}_A(\boldsymbol{R}, \boldsymbol{Y}^k, \boldsymbol{Z}^k)$
4: $\quad \boldsymbol{Z}^{k+\frac{1}{2}} = \boldsymbol{Z}^k + \gamma\beta \cdot \mathcal{P}_{\mathcal{Z}_0}(\boldsymbol{Y}^k - \boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T)$
5: $\quad \boldsymbol{Y}^{k+1} = \underset{\boldsymbol{Y} \in \mathcal{Y}}{\arg\min}\ \mathcal{L}_A(\boldsymbol{R}^{k+1}, \boldsymbol{Y}, \boldsymbol{Z}^{k+\frac{1}{2}})$
6: $\quad \boldsymbol{Z}^{k+1} = \boldsymbol{Z}^{k+\frac{1}{2}} + \gamma\beta \cdot \mathcal{P}_{\mathcal{Z}_0}(\boldsymbol{Y}^{k+1} - \boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T)$
7: **end while**

Note that the standard PRSM updates the dual multipliers with the projection operator $\mathcal{P}_{\mathcal{Z}_0}$ set to the identity operator. The projection on the dual multiplier $\boldsymbol{Z}$ is motivated from the additional information from Theorem 2.2. The algorithm fixes these known elements to be the optimal elements at every iteration. Details of the convergence proof of the rPRSM scheme can be found in Graham et al. (2020, theorem 3.2). The Y-subproblem at line 5 in Algorithm 1 differs from the one in Graham et al. (2020) due to the difference in the gangster constraints in the set $\mathcal{Y}$.

Model (2.10) can be solved by using a standard interior point SDP solver. However, this approach encounters the difficulty that results from maintaining double nonnegativity, that is, $\boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T \geq 0$, $\boldsymbol{R} \geq 0$ at each iteration. This was handled in Burkowski et al. (2014) by adding the most violated cutting planes from the nonnegativity. This approach becomes computationally expensive as the number of cutting planes increases.

The rationale for using a splitting method is as follows. Algorithm 1 can handle the polyhedral and cone constraints $\boldsymbol{Y} \in \mathcal{Y}, \boldsymbol{R} \in \mathcal{R}$ efficiently if we separate this into two problems. The R-subproblem (line 3 in Algorithm 1) concentrates on the positive semidefinite and trace constraints, whereas the Y-subproblem (line 5 in Algorithm 1) handles the interval and gangster constraints. Furthermore, the solutions that we typically look for from the SDP relaxations of hard combinatorial problems are rank one. The rank-one solutions are degenerate points in the sense of Wolkowicz et al. (2000, chapter 3) because the number of linear equality constraints is $n_0 + 1 - p +$

$\sum_{i=1}^{p}(m_i(m_i-1)-2)/2$ in the SDP relaxation. The degeneracy results in ill-conditioned linear systems when computing search directions of interior point methods. Hence, this approach can increase runtime and decrease accuracy. Lastly, $\hat{E}$ typically contains very large elements that arise from the collisions between rotamers. The large discrepancy among elements in $\hat{E}_{i,j}$ causes problems with the performance of typical interior point methods. We discuss how we handle this challenge in Section 4.2.

3.1. Update Formulae

We now present the formulae for the $\boldsymbol{R}$ and $\boldsymbol{Y}$ updates in Algorithm 1. These formulae are discussed (Graham et al. 2020) but are included here for completeness.

3.1.1. *R*-Update. The formula for the R-subproblem, with $\mathcal{L}_A$ defined in (3.1), is as follows:

$$\begin{aligned}
\boldsymbol{R}^{k+1} &= \arg\min_{\boldsymbol{R}\in\mathcal{R}} \ \mathcal{L}_A(\boldsymbol{R},\boldsymbol{Y}^k,\boldsymbol{Z}^k)\\
&= \arg\min_{\boldsymbol{R}\in\mathcal{R}} \left\|\boldsymbol{Y}^k - \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T + \frac{1}{\beta}\boldsymbol{Z}^k\right\|_F^2\\
&= \arg\min_{\boldsymbol{R}\in\mathcal{R}} \left\|\boldsymbol{R} - \boldsymbol{V}^T\left(\boldsymbol{Y}^k + \frac{1}{\beta}\boldsymbol{Z}^k\right)\boldsymbol{V}\right\|_F^2\\
&= \mathcal{P}_{\mathcal{R}}\left(\boldsymbol{V}^T\left(\boldsymbol{Y}^k + \frac{1}{\beta}\boldsymbol{Z}^k\right)\boldsymbol{V}\right)\\
&= \boldsymbol{U}\,\mathrm{Diag}(\mathcal{P}_{\Delta_{p+1}}(\boldsymbol{d}))\,\boldsymbol{U}^T,
\end{aligned}$$

where the second equality holds by completing the square; the third equality holds due to $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$; and the last equality follows from the eigenvalue decomposition

$$\boldsymbol{V}^T\left(\boldsymbol{Y}^k + \frac{1}{\beta}\boldsymbol{Z}^k\right)\boldsymbol{V} = \boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{d})\boldsymbol{U}^T,$$

and $\mathcal{P}_{\Delta_{p+1}}(\cdot)$ is the projection operator onto the simplex $\Delta_{p+1} = \{\boldsymbol{z}\in\mathbb{R}_+^{n_0+1-p} : \bar{\boldsymbol{e}}^T\boldsymbol{z} = 1+p\}$.

3.1.2. *Y*-Update. The update rule for $\boldsymbol{Y}$ is as follows:

$$\begin{aligned}
\boldsymbol{Y}^{k+1} &= \arg\min_{\boldsymbol{Y}\in\mathcal{Y}} \ \mathcal{L}_A(\boldsymbol{R}^{k+1},\boldsymbol{Y},\boldsymbol{Z}^{k+\frac{1}{2}})\\
&= \arg\min_{\boldsymbol{Y}\in\mathcal{Y}} \left\|\boldsymbol{Y} - \left(\boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T - \frac{1}{\beta}(\hat{\boldsymbol{E}} + \boldsymbol{Z}^{k+\frac{1}{2}})\right)\right\|_F^2\\
&= \mathcal{P}_{\mathrm{box}}\left(G_{\hat{\mathcal{J}}^c}\left(\boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T - \frac{1}{\beta}(\hat{\boldsymbol{E}} + \boldsymbol{Z}^{k+\frac{1}{2}})\right)\right),
\end{aligned} \tag{3.2}$$

where $\mathcal{P}_{\mathrm{box}}$ is the projection onto the polyhedral set $\{\boldsymbol{Y}\in\mathbb{S}^{n_0+1} : 0 \le \boldsymbol{Y} \le 1\}$.

3.2. Bounding

In this section, we present some strategies for computing lower and upper bounds to (IQP).

3.2.1. Lower Bounds from Lagrange Relaxation. We now discuss a strategy for computing a valid lower bound to p^*_{IQP}. Exact solutions of the DNN relaxation (2.10) provide lower bounds to (IQP). However, we often terminate algorithms when the stopping criteria are met for a predefined tolerance, and we never set the tolerance to be exactly zero in practice. A near optimal point $\tilde{\boldsymbol{Y}}$ can result in

$$p^*_{\mathrm{DNN}} \le \langle\hat{\boldsymbol{E}},\tilde{\boldsymbol{Y}}\rangle \ \text{ and } \ p^*_{\mathrm{IQP}} < \langle\hat{\boldsymbol{E}},\tilde{\boldsymbol{Y}}\rangle$$

and produce an invalid lower bound to p^*_{IQP}. Hence, we provide a method for computing a *valid lower bound* to (IQP) and avoid this issue.

This follows the approaches in Graham et al. (2020), Oliveira et al. (2018), and Eckstein (2020) and obtains lower bounds via the dual to the DNN relaxation in (2.10). Let the dual functional $g : \mathbb{S}^{n_0+1} \to \mathbb{R}$ be defined as

$$g(\boldsymbol{Z}) := \min_{\boldsymbol{R}\in\mathcal{R},\,\boldsymbol{Y}\in\mathcal{Y}} \langle\hat{\boldsymbol{E}},\boldsymbol{Y}\rangle + \langle\boldsymbol{Z},\boldsymbol{Y} - \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T\rangle.$$

Let $\overline{\boldsymbol{Z}} \in \mathbb{S}^{n_0+1}$ be given. Note that

$$\begin{aligned}\min_{\boldsymbol{R}\in\mathcal{R},\,\boldsymbol{Y}\in\mathcal{Y}} \langle \hat{\boldsymbol{E}}, \boldsymbol{Y}\rangle + \langle \overline{\boldsymbol{Z}}, \boldsymbol{Y} - \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T\rangle &= \min_{\boldsymbol{Y}\in\mathcal{Y}} \langle \hat{\boldsymbol{E}} + \overline{\boldsymbol{Z}}, \boldsymbol{Y}\rangle + \min_{\boldsymbol{R}\in\mathcal{R}} \langle -\boldsymbol{V}^T\overline{\boldsymbol{Z}}\boldsymbol{V}, \boldsymbol{R}\rangle \\ &= \min_{\boldsymbol{Y}\in\mathcal{Y}} \langle \hat{\boldsymbol{E}} + \overline{\boldsymbol{Z}}, \boldsymbol{Y}\rangle - (p+1)\lambda_{\max}(\boldsymbol{V}^T\overline{\boldsymbol{Z}}\boldsymbol{V}),\end{aligned}$$

where $\lambda_{\max}$ is the maximum eigenvalue function. Hence, we compute a valid lower bound to the optimal value p^*_{DNN} of the model (2.10) by using weak duality:

$$p^*_{\mathrm{DNN}} = \max_{\boldsymbol{Z}} g(\boldsymbol{Z}) \geq g(\boldsymbol{Z}) = \min_{\boldsymbol{Y}\in\mathcal{Y}} \langle \hat{\boldsymbol{E}} + \boldsymbol{Z}, \boldsymbol{Y}\rangle - (p+1)\lambda_{\max}(\boldsymbol{V}^T\boldsymbol{Z}\boldsymbol{V}),$$

where the first equality holds since the constraint qualification holds for Model (2.10). The computation for $\min_{\boldsymbol{Y}\in\mathcal{Y}} \langle \hat{\boldsymbol{E}} + \boldsymbol{Z}, \boldsymbol{Y}\rangle$ is inexpensive.

3.2.2. Upper Bounds from Nearest Binary Feasible Solutions. We now present two strategies for computing upper bounds to the SCP problem. These strategies are derived from those presented in Burkowski et al. (2014), and we include them here for completeness. We obtain upper bounds by finding feasible solutions to the original integer model in (2.3). Let $(\boldsymbol{R}^{\mathrm{out}}, \boldsymbol{Y}^{\mathrm{out}}, \boldsymbol{Z}^{\mathrm{out}})$ be the output of the algorithm.

1. Let $\boldsymbol{x}^{\mathrm{approx}} \in \mathbb{R}^{n_0}$ be the second through to the last elements of the first column of $\boldsymbol{Y}^{\mathrm{out}}$. Note that $0 \leq \boldsymbol{x}^{\mathrm{approx}} \leq 1$. Then the nearest feasible solution to (IQP) from $\boldsymbol{x}^{\mathrm{approx}}$ can be found by solving the following projection:

$$\min_{\boldsymbol{x}} \{\|\boldsymbol{x} - \boldsymbol{x}^{\mathrm{approx}}\|^2 \,:\, \boldsymbol{A}\boldsymbol{x} = \overline{\boldsymbol{e}}_p,\, \boldsymbol{x} \in \{0,1\}^{n_0}\}. \tag{3.3}$$

It is shown in Burkowski et al. (2014) that solving (3.3) is equivalent to solving the following *linear program*:

$$\min_{\boldsymbol{x}} \{\langle \boldsymbol{x}, \boldsymbol{x}^{\mathrm{approx}}\rangle \,:\, \boldsymbol{A}\boldsymbol{x} = \overline{\boldsymbol{e}}_p,\, \boldsymbol{x} \geq 0\}. \tag{3.4}$$

2. We now let $\boldsymbol{x}^{\mathrm{approx}}$ be the second through to the last elements of the most dominant eigenvector of $\boldsymbol{Y}^{\mathrm{out}}$. Note that we again have $0 \leq \boldsymbol{x}^{\mathrm{approx}} \leq 1$ by the Perron-Frobenius theorem. We again obtain the nearest feasible solution to $\boldsymbol{x}^{\mathrm{approx}}$ by solving (3.4).

Remark 3.1. In fact, solving (3.4) does not require using any linear program software; we can obtain the optimal solution for (3.4) as follows. We partition $\boldsymbol{x}^{\mathrm{approx}}$ into p subvectors of sizes $m_i = |\mathcal{V}_i|$, for $i = 1, \ldots, p$. Let $\boldsymbol{x}^i \in \mathbb{R}^{m_i}$ be the subvector of $\boldsymbol{x}^{\mathrm{approx}}$ associated with ith rotamer set $\mathcal{V}_i$, that is, $\boldsymbol{x}^{\mathrm{approx}} = [\boldsymbol{x}^1; \boldsymbol{x}^2; \ldots; \boldsymbol{x}^p]$. We define $\hat{\boldsymbol{x}}^i \in \mathbb{R}^{m_i}$ as follows:

$$\hat{\boldsymbol{x}}^i_j = \begin{cases} 1, & \text{if } \boldsymbol{x}^i_j = \max\limits_{\ell\in[m_i]} \{\boldsymbol{x}^i_\ell\} \\ 0, & \text{otherwise.}\end{cases}$$

If there is a subvector $\hat{\boldsymbol{x}}^i$ with more than one 1 in its components, we pick only one 1 and set the remaining to be zero. We then form $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{x}}^1; \hat{\boldsymbol{x}}^2; \ldots; \hat{\boldsymbol{x}}^p] \in \mathbb{R}^{n_0}$. It is clear that $\hat{\boldsymbol{x}}$ is feasible for (2.1). We use $\hat{\boldsymbol{x}}^T\boldsymbol{E}\hat{\boldsymbol{x}}$ as an upper bound to the SCP problem.

4. Computational Experiments for Algorithm 1 Using Real-World Data

Section 4.1 presents the parameter settings and stopping criteria. Section 4.2 explains how to process the data from the PDB to obtain the energy matrix $\boldsymbol{E}$. Section 4.3 presents the numerical results using rPRSM and shows, using the bounding strategies presented in Section 3.2, that we provably solve many instances to optimality.

4.1. Stopping Criteria and Parameter Settings

4.1.1. Stopping Criteria. We terminate rPRSM when either of the following conditions is satisfied.

1. Maximum number of iterations, denoted by "maxiter" is achieved.
2. For a given tolerance ϵ, the following bound on the primal and dual residuals holds for s_t sequential times:

$$\max\left\{\frac{\|\boldsymbol{Y}^k - \boldsymbol{V}\boldsymbol{R}^k\boldsymbol{V}^T\|_F}{\|\boldsymbol{Y}^k\|_F},\, \beta\|\boldsymbol{Y}^k - \boldsymbol{Y}^{k-1}\|_F\right\} < \epsilon.$$

3. Let $\{l_1,\ldots,l_k\}$ and $\{u_1,\ldots,u_k\}$ be sequences of lower and upper bounds discussed in Sections 3.2.1 and 3.2.2, respectively. Any of the lower bounds achieve the best upper bound, that is,

$$\max\{l_1,\ldots,l_k\} \geq \min\{u_1,\ldots,u_k\}.$$

4.1.2. Parameter Settings. We use the following parameters related to the implementation of Algorithm 1:

$$\beta = \max\{\lfloor 0.5 * n_0/p \rfloor, 1\}, \ \gamma = 0.99.$$

The parameters related to stopping criteria are

$$\text{maxiter} = p(n_0+1) + 10^4, \quad \epsilon = 10^{-10}, \quad s_t = 100.$$

For the initial iterates for rPRSM, we use

$$\boldsymbol{Y}^0 = 0, \ \boldsymbol{Z}^0 = \mathcal{P}_{\mathcal{Z}_A}(\boldsymbol{Y}^0).$$

4.2. Energy Matrix Computation

We now describe the process for constructing the energy matrix E. We use a Python script that is run as an extension of the University of California, San Francisco (UCSF) Chimera[5] application. A detailed implementation can be found in Burkowski (2015, chapter 7). The protein data files were taken from the PDB to obtain the coordinates of all atoms in the protein. To get the energy values required by the algorithm, the native side chain conformations were replaced by rotamers extracted from a rotamer library provided by the Dunbrack Laboratory (Dunbrack and Karplus 1993).

Some approaches use an energy evaluation based on a piece-wise linear approximation of the Lennard-Jones potential formula (Canutescu et al. 2003, Xu and Berger 2006). Here, we used the Lennard-Jones potential formula, which provides a more accurate energy value computation. In brief, the Lennard-Jones potential formula takes the Euclidean distance between a pair of atoms in combination with certain parameters that are chosen dependent on the type of amino acids. A more detailed explanation of these energy computations can be found in Burkowski (2015, chapters 6 and 7).

To complete the process, we used a strategy (called *dead end elimination*) to reduce the size of the rotamer sets associated with each amino acid. The simple idea behind this strategy is that a rotamer can be removed from its rotamer set if there is another rotamer in that set that gives a better energy value regardless of the rotamer selections for the neighboring amino acids. From the various approaches for the dead end elimination, we followed Goldstein's criteria (Goldstein 1994).

Let $\mathcal{U}$ be a side-chain conformation of a protein. The energy of the conformation $\mathcal{U}$ is

$$E(\mathcal{U}) = \sum_{i=1}^{n_0} E_{\text{self}}(u_i) + \sum_{i=1}^{n_0-1}\sum_{j=i+1}^{n_0} E_{\text{pair}}(u_i,u_j),$$

where u_i is a side-chain conformation of an amino acid, $E_{\text{self}}(u_i)$ is the energy corresponding to u_i and the backbone, and $E_{\text{pair}}(u_i,u_j)$ is the energy formed by u_i and u_j, a rotamer associated with a neighboring amino acid. In our formulation, we placed $E_{\text{self}}(u_i)$ along the diagonal of E and $E_{\text{pair}}(u_i,u_j)$ on the appropriate off-diagonal positions of E as shown in Section 2.1.

4.2.1. Removing Collisions. We typically observe that $\boldsymbol{E}$ contains some very large elements of the order of $\boldsymbol{E}_{i,j} \approx 10^{10}$. This arises due to collisions between rotamers. $\boldsymbol{E}_{i,j} \gg 0$ that are often greater than 10^{10} can be seen from the Lennard-Jones potential formula that small Euclidean distances between two distinct rotamers as part of the denominators of fractions.

In general, having a very large spread of values in data, bad scaling, results in numerical instabilities. The matrix $\boldsymbol{E}$ often has elements that are of the order $O(10^{10})$, as well as elements that are of the order $O(1)$. When there is a large discrepancy among the elements of $\boldsymbol{E}$, scaling $\boldsymbol{E}$ would make the relatively small values close to zero and lead to loss of precision in the solution. However, this ill-posed data do not take place as a problem in our implementation. Recall that we update the $\boldsymbol{Y}$ iterate (3.2) as follows:

$$\begin{aligned}\boldsymbol{Y}^{k+1} &= \mathcal{P}_{\mathcal{Y}}\left(G_{\hat{\mathcal{Y}}^c}\left(\boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T - \frac{1}{\beta}(\hat{\boldsymbol{E}} + \boldsymbol{Z}^{k+\frac{1}{2}})\right)\right)\\ &= \mathcal{P}_{\mathcal{Y}}\left(G_{\hat{\mathcal{Y}}^c}\left(-\frac{1}{\beta}\hat{\boldsymbol{E}} + \left[\boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T - \frac{1}{\beta}\boldsymbol{Z}^{k+\frac{1}{2}}\right]\right)\right).\end{aligned}$$

For simplicity, we let $\boldsymbol{T} := -\frac{1}{\beta}\hat{\boldsymbol{E}} + [\boldsymbol{V}\boldsymbol{R}^{k+1}\boldsymbol{V}^T - \frac{1}{\beta}\boldsymbol{Z}^{k+\frac{1}{2}}]$. If the $(\hat{i},\hat{j})$th element of $\hat{\boldsymbol{E}} = \mathrm{BlkDiag}(0,\boldsymbol{E})$ is very large, the projection $\mathcal{P}_{\mathcal{Y}}$ sets the $(\hat{i},\hat{j})$-element of $\boldsymbol{T}$ to zero beecause $\boldsymbol{T}_{\hat{i},\hat{j}} \ll 0$. Hence, for those positions $(\hat{i},\hat{j})$ with very large energy values, the constraint $\boldsymbol{Y}_{\hat{i},\hat{j}} = 0$ is implicitly imposed. We can interpret this as having implicit gangster constraints on these elements. Consequently, the large elements do not contribute to the objective value because $\hat{\boldsymbol{E}}_{\hat{i},\hat{j}}\boldsymbol{Y}_{\hat{i},\hat{j}} = 0$.

We can also take advantage of large values in the data to increase the number of the gangster indices (eliminate edges in the graph).

Lemma 4.1. *Let $\boldsymbol{x}$ be any feasible point for* (IQP) *and let $f(x) = \boldsymbol{x}^T\boldsymbol{E}\boldsymbol{x}$ be its objective value. Let $N_E = \sum_{\{(i,j):E_{(i,j)}<0\}}\boldsymbol{E}_{i,j}$ and suppose that*

$$\boldsymbol{E}_{i_0,j_0} > f(x) - N_E, \text{ for some } i_0, j_0$$

holds. Then for any optimal solution $\boldsymbol{x}^*$ to (IQP)*, we have $\boldsymbol{x}^*_{i_0}\boldsymbol{x}^*_{j_0} = 0$.*

Proof. Let $\boldsymbol{x}^*$ be an optimal solution to (IQP). Let U^* be the set of indices formed by the positive entries of $\begin{pmatrix}1\\ \boldsymbol{x}^*\end{pmatrix}\begin{pmatrix}1\\ \boldsymbol{x}^*\end{pmatrix}^T$. We note that, for any index set S, we have

$$\sum_{(i,j)\in S}\boldsymbol{E}_{i,j} = \sum_{(i,j)\in S\cap\{(i,j):\boldsymbol{E}_{i,j}\geq 0\}}\boldsymbol{E}_{i,j} + \sum_{(i,j)\in S\cap\{(i,j):E_{i,j}<0\}}\boldsymbol{E}_{i,j} \geq 0 + N_E = N_E.$$

Supposed to the contrary that $\boldsymbol{x}^*$ holds $\boldsymbol{x}^*_{i_0}\boldsymbol{x}^*_{j_0} = 1$, that is, $\boldsymbol{x}^*_{i_0} = \boldsymbol{x}^*_{j_0} = 1$. Then we reach the following contradiction:

$$p^*_{\mathrm{IQP}} = \langle \boldsymbol{x}^*, \boldsymbol{E}\boldsymbol{x}^*\rangle = \boldsymbol{E}_{i_0,j_0} + \left(\boldsymbol{E}_{i_0,j_0} + \sum_{(i,j)\in U^*\setminus\{(i_0,j_0)\}}\boldsymbol{E}_{i,j}\right) \geq \boldsymbol{E}_{i_0j_0} + N_E > f(\boldsymbol{x}). \quad \square$$

Corollary 4.1. *Let i_0 be an index such that $\boldsymbol{E}_{i_0,i_0} > f(\boldsymbol{x}) - N_E$, where $f(\boldsymbol{x}), N_E$ defined in Lemma 4.1. Then, for any optimal solution $\boldsymbol{x}^*$ to* (IQP)*, we have*

$$\boldsymbol{Y}_{\boldsymbol{x}^*} := \begin{pmatrix}1\\ \boldsymbol{x}^*\end{pmatrix}\begin{pmatrix}1\\ \boldsymbol{x}^*\end{pmatrix}^T \in \{\boldsymbol{Y}\in\mathbb{S}^{n_0+1} : \boldsymbol{Y}(:,i_0) = 0,\ \boldsymbol{Y}(i_0,:) = 0\}.$$

Proof. Let i_0 be an index such that $E_{i_0,i_0} > f(\boldsymbol{x}) - N_E$. Then $\boldsymbol{x}^*_{i_0} = 0$ by Lemma 4.1. We note that $\boldsymbol{Y}_{\boldsymbol{x}^*}$ is a positive semidefinite matrix. If a diagonal entry of a positive semidefinite is zero, then its corresponding column and row must be zero. $\square$

By Lemma 4.1 and Corollary 4.1, if we detect entries i_0, j_0 with the property $\boldsymbol{E}_{i_0,j_0} > f(\boldsymbol{x}) - N_E$, then we may strengthen the model by adding the constraints

$$\mathcal{K} = \left\{\boldsymbol{Y}\in\mathbb{S}^{n_0+1} : \begin{array}{ll}\boldsymbol{Y}(i_0,j_0) = \boldsymbol{Y}(j_0,i_0) = 0, & \text{for } i_0\neq j_0 \text{ such that } \boldsymbol{E}_{i_0,j_0} > f(\boldsymbol{x}) - N_E\\ \boldsymbol{Y}(:,i_0) = 0,\ \boldsymbol{Y}(i_0,:) = 0, & \text{for } i_0 \text{ such that } \boldsymbol{E}_{i_0,i_0} > f(\boldsymbol{x}) - N_E\end{array}\right\}.$$

This can be easily realized by adding more members to the gangster index set $\hat{\mathcal{J}}$.

4.3. Experiments with Real-World Data

In this section, we provide numerical experiments with real-world data from the PDB and discuss the strengths of the DNN relaxation. The data and codes used in the experiments are available at Burkowski et al. (2024). The empirics clearly illustrate the strengths of the DNN relaxation. In addition, this approach avoids the numerical instabilities that can originate from the large positive values in the data matrix E. The empirics also show that the DNN relaxation provides a significant improvement to just using the SDP relaxation.

Table 1. Computational Results on Selected Protein Data Bank Instances

Problem data				Numerical results			Timing	
No.	Name	p	n_0	lbd	ubd	rel-gap	Iterations	Time (s)
10	2IGD	50	126	−78.50608	−78.50608	5.39611 e-15	500	19.43
20	1VQB	75	406	−96.94940	−96.94940	4.34568 e-14	900	179.35
30	2ACY	84	580	−146.32254	−146.32254	1.06468 e-14	7,800	2,610.24
40	2TGI	100	355	−14.03554	−14.03554	2.46249 e-13	1,300	136.30
50	2SAK	111	214	−239.86975	−239.86975	1.08995 e-12	500	25.50
60	2CPL	132	819	−284.97180	−284.97180	9.75693 e-15	5,900	3,292.98
70	1CV8	146	730	−213.13554	−213.13554	3.28738 e-13	5,600	2,572.99
80	2ENG	162	867	82.01797	82.01797	1.33295 e-13	14,200	8,274.48
90	1A7S	179	524	−239.78218	−239.78218	1.00542 e-14	1,200	314.57
100	1MRJ	208	1,178	−295.13711	−295.13711	1.70740 e-13	2,300	2,421.15
110	1EZM	239	1,497	−217.36581	−217.36581	3.49620 e-13	2,300	3,876.18
120	1SBP	256	1,704	−271.08838	−271.08838	3.59996 e-14	40,000	609,487.29
130	3PTE	284	2,006	161.17216	161.17216	5.09815 e-15	13,500	250,604.17

We select instances listed in Canutescu et al. (2003) with proteins that have up to 300 amino acids. All instances in Table 1 are tested using MATLAB version 2021a on Dell XPS 8940 with 11th Gen Intel(R) Core(TM) i5-11400 @ 2.60 GHz with 32 GB memory. The following list defines the column headers used in Table 1; the same headers are used in the additional numerical experiments that are displayed in Section A of the Online Appendix.

1. name: instance name;
2. p: the number of amino acids;
3. n_0: the total number of rotamers;
4. lbd: the lower bound obtained by running rPRSM;
5. ubd: the upper bound obtained by running rPRSM;
6. rel-gap: relative gap of each instance using rPRSM, where

$$\text{relative gap} := 2\,\frac{|\text{best feasible upper bound} - \text{best lower bound}|}{|\text{best feasible upper bound} + \text{best lower bound} + 1|};$$

7. iterations: number of iterations used by rPRSM with tolerance $\epsilon = 10^{-10}$;
8. time (s): CPU time (in seconds) used by rPRSM.

4.3.1. Discussion. We observe from the relative gap (rel-gap) column of Table 1 that many instances are solved to near machine precision; that is, most of the instances display relative gaps that are essentially zero. We recall from Section 3.2.2 that we obtain the upper bounds via finding feasible solutions to (IQP). That we have the relative gap essentially zero certifies the attainment of the *globally optimal* solutions to the SCP problem. Approaches involving heuristic algorithms do not provide a natural means of certifying optimality, relying solely on a comparison of the rotameric solution with naive χ_1 and χ_2 angles from the PDB while ignoring optimality of the discretized solution. We highlight that we provide not only the global optimal solutions but also a way to certify their optimality.

4.3.2. Tighter Relaxation. We illustrate the strengths of the DNN relaxation by computing the near optimal values of the DNN relaxation and the SDP relaxation. In our test, we selected five small instances. As discussed above, some elements of the energy matrix E are typically very large due to the collisions in rotamers, typically at least 10 digits. These cause numerical difficulties when a standard interior point solver is used. Hence, in our test, we set the entries $\boldsymbol{E}_{i,j} = \min\{10^4, \boldsymbol{E}_{i,j}\}$, $\forall i,j$, in order to avoid the difficulties from having these large elements. The rPRSM is used for DNN relaxation, and SDPT3[6] is used for solving the SDP relaxation.

The displayed values in Table 2 are the best lower bounds found from the rPRSM and the optimal values reported by SDPT3. The DNN relaxation clearly shows superior performance over the SDP relaxation as shown by the larger values.

Table 2. Solver Optimal Values of the DNN and SDP Relaxations on Selected Instances

Problem no.	Instance	DNN relaxation	SDP relaxation
1	1AIE	−46.96	−2,460.53
2	2ERL	55.33	−18,241.26
3	1CBN	−40.43	−22,380.58
4	1RB9	−76.97	−23,936.35
5	1BX7	16.96	−23,965.88

5. Conclusions

We presented a simplified way of formulating a relaxation of the SCP problem. The SCP problem was first formulated as an IQP, and then the facially reduced SDP relaxation was derived. We then identified redundant constraints to the IQP to complete the facially reduced DNN relaxation. FR allowed for a natural splitting of the variables and provided an excellent environment for using splitting methods. We applied the rPRSM to solve the DNN relaxation of the SCP problem. The efficiency and accuracy of our approach are illustrated in the numerical experiments using data from the Protein Data Bank. In particular, we provably found the optimal solutions to many of the instances that we chose from the Protein Data Bank.

Endnotes

[1] See https://www.rcsb.org/.

[2] Of 131 test problems, one problem had a positive gap; five other problems had gaps of approximately 10^{-6}.

[3] $\mathcal{V}_i$ indicates the ith rotamer set and v_i^j indicates the jth candidate in the ith rotamer set $\mathcal{V}_i$.

[4] The UCSF Chimera software can be found at https://www.cgl.ucsf.edu/chimera/download.html.

[5] The UCSF Chimera software can be found at https://www.cgl.ucsf.edu/chimera/download.html.

[6] See https://www.math.cmu.edu/~reha/sdpt3.html, version SDPT3 4.0 (Toh et al. 1999).

References

Akutsu T (1997) NP-hardness results for protein side-chain packing. *Genome Informatics* 8:180–186.

Althaus E, Kohlbacher O, Lenhof HP, Mauller P (2002) A combinatorial approach to protein docking with flexible side chains. *J. Comput. Biology* 9(4):597–612.

Bahadur D, Akutsu T, Tomita E, Seki T (2004) Protein side-chain packing problem: A maximum edge-weight clique algorithmic approach. *J. Bioinform. Comput. Biol.* 3(1):103–126.

Bower M, Cohen F, Dunbrack R (1997) Prediction of protein side-chain rotamers from a backbone-dependent rotamer library: A new homology modeling tool. *J. Molecular Biology* 267(5):1268–1282.

Burkowski F (2015) *Computational and Visualization Techniques for Structural Bioinformatics Using Chimera*, Chapman & Hall/CRC Mathematical and Computational Biology Series (Chapman and Hall/CRC, London).

Burkowski F, Cheung YL, Wolkowicz H (2014) Efficient use of semidefinite programming for selection of rotamers in protein conformations. *INFORMS J. Comput.* 26(4):748–766.

Burkowski F, Im H, Wolkowicz H (2024) Repository to "A restricted peacement-rachform splitting method for protein side-chain positioning problem". http://dx.doi.org/10.1287/ijoc.2023.0094.cd, https://github.com/INFORMSJoC/2023.0094.

Canutescu A, Shelenkov A, Dunbrack R (2003) A graph-theory algorithm for rapid protein side-chain prediction. *Protein Sci.* 12(9):2001–2014.

Chazelle B, Kingsford C, Singh M (2004) A semidefinite programming approach to side chain positioning with new rounding strategies. *INFORMS J. Comput.* 16(4):380–392.

Desmet J, Maeyer MD, Hazes B, Lasters I (1992) The dead-end elimination theorem and its use in protein side-chain positioning. *Nature (London)* 356(6369):539–542.

Drusvyatskiy D, Wolkowicz H (2017) The many faces of degeneracy in conic optimization. *Foundations Trends Optim.* 3(2):77–170.

Dunbrack R Jr, Karplus M (1993) Backbone-dependent rotamer library for proteins application to side-chain prediction. *J. Molecular Biology* 230(2):543–574.

Eckstein J (2020) Deriving solution value bounds from the ADMM. *Optim. Lett.* 14:1289–1303.

Eriksson O, Zhou Y, Elofsson A (2001) Side chain-positioning as an integer programming problem. *Algorithms Bioinformatics*, Lecture Notes in Computational Science, vol. 2149 (Springer, Berlin), 128–141.

Goldstein R (1994) Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophys. J.* 66(5):1335–1340.

Graham N, Hu H, Im H, Li X, Wolkowicz H (2020) A restricted dual Peaceman-Rachford splitting method for QAP. Technical report, University of Waterloo, Waterloo, ON.

Holm L, Sander C (1991) Database algorithm for generating protein backbone and side-chain co-ordinates from a C^{α} trace: Application to model building and detection of co-ordinate errors. *J. Molecular Biology* 218(1):183–194.

Kingsford C, Chazelle B, Singh M (2005) Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics (Oxford, England)* 21(7):1028–1039.

Laudet V, Gronemeyer H (2002) 3-ligand binding. Laudet V, Gronemeyer H, eds. *The Nuclear Receptor FactsBook* (Academic Press, London), 37–41.

Lee C (1994) Predicting protein mutant energetics by self-consistent ensemble optimization. *J. Molecular Biology* 236(3):918–939.
Li X, Pong T, Sun H, Wolkowicz H (2019) A strictly contractive Peaceman-Rachford splitting method for the doubly nonnegative relaxation of the minimum cut problem. Technical report, University of Waterloo, Waterloo, ON.
Looger L, Dwyer M, Smith J, Hellinga H (2003) Computational design of receptor and sensor proteins with novel functions. *Nature (London)* 423(6936):185–190.
Marze N, Roy-Burman S, Sheffler W, Gray J (2018) Efficient flexible backbone protein-protein docking for challenging targets. *Computer Appl. Biosci.* 34(20):3461–3469.
Oliveira D, Wolkowicz H, Xu Y (2018) ADMM for the SDP relaxation of the QAP. *Math. Program. Comput.* 10(4):631–658.
Samudrala R, Moult J (1998) Determinants of side chain conformational preferences in protein structures. *Protein Engrg.* 11(11):991–997.
Shenkin P, Farid H, Fetrow J (1996) Prediction and evaluation of side-chain conformations for protein backbone structures. *Proteins: Structure, Function, Bioinform.* 26(3):323–352.
Toh K, Todd M, Tütüncü R (1999) SDPT3—A MATLAB software package for semidefinite programming, version 1.3. *Optim. Methods Software* 11/12(1–4):545–581.
Wang C, Bradley P, Baker D (2007) Protein-protein docking with backbone flexibility. *J. Molecular Biology* 373(2):503–519.
Wolkowicz H, Saigal R, Vandenberghe L, eds. (2000) *Handbook of Semidefinite Programming,* International Series in Operations Research & Management Science, vol. 27 (Kluwer Academic Publishers, Boston).
Xu J, Berger B (2006) Fast and accurate algorithms for protein side-chain packing. *J. ACM* 53(4):533–557.