HPR-QP: A dual Halpern Peaceman–Rachford method for solving large-scale convex composite quadratic programming*

Kaihuang Chen† Defeng Sun‡ Yancheng Yuan§ Guojun Zhang¶
Xinyuan Zhao‖

July 4, 2025

Abstract

In this paper, we introduce HPR-QP, a dual Halpern Peaceman–Rachford (HPR) method designed for solving large-scale convex composite quadratic programming. One distinctive feature of HPR-QP is that, instead of working with the primal formulations, it builds on the novel restricted Wolfe dual introduced in recent years. It also leverages the symmetric Gauss–Seidel technique to simplify subproblem updates without introducing auxiliary slack variables that typically lead to slow convergence. By restricting updates to the range space of the Hessian of the quadratic objective function, HPR-QP employs proximal operators of smaller spectral norms to speed up the convergence. Shadow sequences are elaborately constructed to deal with the range space constraints. Additionally, HPR-QP incorporates adaptive restart and penalty parameter update strategies, derived from the HPR method's $O(1/k)$ convergence in terms of the Karush–Kuhn–Tucker residual, to further enhance its performance and robustness. Extensive numerical experiments on benchmark data sets using a GPU demonstrate that our Julia implementation of HPR-QP significantly outperforms state-of-the-art solvers in both speed and scalability.

Keywords: Convex composite quadratic programming, Halpern Peaceman–Rachford method, Restricted Wolfe dual, Symmetric Gauss-Seidel

MSCcodes: 90C20, 90C06, 90C25, 65Y20

1 Introduction

In this paper, we develop a dual Halpern Peaceman–Rachford (HPR) method with semi-proximal terms [43] for solving the large-scale convex composite quadratic programming (CCQP) problem:

$$\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\langle x,Qx\rangle+\langle c,x\rangle+\phi(x)\,\middle|\,Ax\in\mathcal{K}\right\},\tag{1.1}$$

*The work of Defeng Sun was supported by the Research Center for Intelligent Operations Research, RGC Senior Research Fellow Scheme No. SRFS2223-5S02, and GRF Project No. 15307822. The work of Yancheng Yuan was supported by the Research Center for Intelligent Operations Research. The work of Xinyuan Zhao was supported in part by the National Natural Science Foundation of China under Project No. 12271015.

†Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, (kaihuang.chen@connect.polyu.hk).

‡Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, (defeng.sun@polyu.edu.hk).

§Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, (yancheng.yuan@polyu.edu.hk).

¶Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, (guojun.zhang@connect.polyu.hk).

‖Department of Mathematics, Beijing University of Technology, Beijing, P.R. China, (xyzhao@bjut.edu.cn).

where $Q : \mathbb{R}^n \to \mathbb{R}^n$ is a self-adjoint positive semidefinite linear operator, $c \in \mathbb{R}^n$ is a given vector, and $\phi : \mathbb{R}^n \to (-\infty, +\infty]$ is a proper, closed, and convex function. Here, $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator, and $\mathcal{K} := \{y \in \mathbb{R}^m \mid -\infty \le l_i \le y_i \le u_i \le +\infty, 1 \le i \le m\}$ is a simple polyhedral set. A key feature of our approach is that it does not require an explicit matrix representation of Q, which makes the proposed method particularly suitable for large-scale or matrix-free settings—e.g., when Q is defined implicitly via Kronecker products or structured operators [1]. In particular, CCQP includes the classical convex QP (CQP):

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2}\langle x, Qx\rangle + \langle c, x\rangle + \delta_{\mathcal{C}}(x) \,\middle|\, Ax \in \mathcal{K} \right\} \tag{1.2}$$

as an important special case, where $\delta_{\mathcal{C}}(\cdot)$ is the indicator function of the box constraint set $\mathcal{C} = \{x \in \mathbb{R}^n \mid L \le x \le U\}$, with $L \in (\mathbb{R} \cup \{-\infty\})^n$ and $U \in (\mathbb{R} \cup \{+\infty\})^n$.

For solving the CQP (1.2), commercial solvers such as Gurobi [17] and CPLEX [21] typically rely on active set methods or interior-point methods. While these methods are robust and effective for small- to medium-sized problems, they face significant scalability challenges when applied to large-scale instances. In particular, interior-point methods suffer from high per-iteration computational costs and substantial memory demands. On the other hand, active set methods—while benefiting from cheaper per-iteration costs—are inherently sequential, making them difficult to parallelize. Even GPU-accelerated interior-point implementations such as CuClarabel [16, 10], which exploit mixed-precision arithmetic, still rely on direct factorization routines, limiting their scalability on large-scale problems.

To overcome these limitations, a variety of first-order solvers, such as SCS [37, 36], OSQP [42], PDQP [33], ABIP [29, 11], PDHCG [20], and PDCS [30], has been developed for large-scale CQP problems. In particular, SCS [37, 36] applies the Douglas–Rachford method [31] to solve convex conic programming with convex quadratic objective functions, while OSQP [42] implements a generalized ADMM [13] tailored for CQP problems. Both solvers support indirect methods for solving the linear systems that arise at each iteration, which improves scalability over direct factorization routines. Recently, Lu and Yang [33] proposed PDQP, which combines the accelerated primal-dual hybrid gradient method [9] with adaptive step sizes and restart strategies—algorithmic enhancements used in the award-winning solver PDLP [2, 3, 32]. Notably, each step of PDQP has explicit update formulas, eliminating the need for linear system solves. Experiments in [33] show that PDQP, implemented in Julia with GPU support, outperforms SCS (on GPU and CPU) and OSQP (on CPU) on large-scale synthetic CQP problems.

Beyond these developments, some theoretical and algorithmic advances have been made in accelerating the (semi-proximal) Peaceman–Rachford (PR) method [47, 45, 43, 46] using the Halpern iteration [18, 41, 28]. Particularly, Zhang et al. [47] developed the HPR method without proximal terms by applying the Halpern iteration to the PR method [13, 31], achieving an $O(1/k)$ iteration complexity in terms of the Karush–Kuhn–Tucker (KKT) residual and the objective error. Sun et al. [43] reformulated the semi-proximal PR method as a degenerate proximal point method (dPPM) [5] with a positive semidefinite preconditioner, and applied the Halpern iteration to derive the HPR method with semi-proximal terms, which also enjoys an $O(1/k)$ iteration complexity. Building on this, Chen et al.[8] introduced HPR-LP, a GPU-accelerated solver for large-scale linear programming (LP), which demonstrated significantly better performance than the award-winning solver PDLP [2, 3, 32]. Given PDQP's strong performance on CQP problems as an extension of PDLP, and HPR-LP's superior results over PDLP on LP tasks, we are motivated to develop a GPU-accelerated HPR method for solving large-scale CCQP problems, including CQP problems.

A natural approach is to apply the HPR method directly to the primal CCQP (1.1) by introducing a single auxiliary slack variable s:

$$\min_{(x,s) \in \mathbb{R}^n \times \mathbb{R}^m} \left\{ \frac{1}{2}\langle x, Qx\rangle + \langle c, x\rangle + \phi(x) + \delta_{\mathcal{K}}(s) \,\middle|\, Ax = s \right\}. \tag{1.3}$$

To simplify the subproblem solving within the HPR framework, this approach typically requires a large proximal operator of the form

$$\mathcal{S}_x = \lambda_Q I_n - Q + \sigma\big(\lambda_A I_n - A^* A\big),$$

where $\lambda_Q \geq \lambda_1(Q)$ and $\lambda_A \geq \lambda_1(A^*A)$ are constants ensuring that $\mathcal{S}_x$ is positive semidefinite, and $\sigma > 0$ is a penalty parameter; $\lambda_1(\cdot)$ denotes the largest eigenvalue of a self-adjoint linear operator. However, the resulting large spectral norm $\|\sqrt{\mathcal{S}_x}\|$ can significantly slow convergence (see Appendix A.1 for algorithmic details). To decouple A and Q (so that each can be handled separately), one may introduce a second auxiliary variable v:

$$\min_{(x,s,v)\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n} \left\{ \frac{1}{2}\langle v, Qv\rangle + \langle c, x\rangle + \phi(x) + \delta_{\mathcal{K}}(s) \;\middle|\; Ax = s,\ x = v \right\}. \tag{1.4}$$

This yields two simpler proximal operators:

$$\mathcal{S}_v = \lambda_Q I_n - Q, \qquad \mathcal{S}_x = \sigma(\lambda_A I_n - A^*A),$$

but the combined spectral radius $\|\sqrt{\mathcal{S}_v}\| + \|\sqrt{\mathcal{S}_x}\|$ remains large, limiting convergence speed (see Appendix A.2 for algorithmic details).

In this work, instead of working on the primal forms, we pay our attention to a novel restricted Wolfe dual of problem (1.1), as recently introduced in [25]:

$$\min_{(y,w,z)\in\mathbb{R}^m\times\mathbb{R}^n\times\mathbb{R}^n} \left\{ \frac{1}{2}\langle w, Qw\rangle + \delta^*_{\mathcal{K}}(-y) + \phi^*(-z) \;\middle|\; -Qw + A^*y + z = c, \quad w \in \mathcal{W} \right\}, \tag{1.5}$$

where $\mathcal{W} := \mathrm{Range}(Q)$, the range space of Q, is explicitly imposed in the constraints, as opposed to the classical Wolfe dual form [12, 44] with $\mathcal{W}$ to be taken as the whole space $\mathbb{R}^n$. Since Q is positive definite on $\mathcal{W}$, one may consider applying a convergent three-block semi-proximal ADMM [22] to solve problem (1.5). However, its convergence guarantee requires choosing the penalty parameter σ proportional to the smallest positive eigenvalue of Q, which leads to slow practical performance. To overcome this limitation, we propose HPR-QP, a dual HPR method to solve problem (1.5). The main features of HPR-QP are summarized below:

1. By leveraging the symmetric Gauss–Seidel (sGS) technique [23, 26], HPR-QP decouples operators A and Q without introducing auxiliary slack variables. Furthermore, restricting updates to $\mathcal{W}$ allows HPR-QP to employ proximal operators with significantly smaller spectral norms for handling Q, thereby accelerating convergence. Moreover, shadow sequences are constructed to address the numerical challenges introduced by subspace constraints. Together, these innovations eliminate the need for the large proximal terms required by primal formulations, offering both theoretical and computational advantages. Numerical results in Section 4 confirm that our restricted Wolfe dual approach substantially outperforms its primal counterparts.

2. HPR-QP incorporates adaptive restart and penalty parameter update strategies, derived from the HPR method's $O(1/k)$ iteration complexity in terms of the KKT residual, to further enhance its performance and robustness. Also, HPR-QP does not require an explicit matrix representation of Q, allowing it to handle extremely large-scale problem instances. For example, it can efficiently solve CQP relaxations of quadratic assignment problems (QAPs) involving up to 8,192 locations.

3. Extensive numerical experiments on benchmark data sets using a GPU demonstrate that our Julia implementation of HPR-QP significantly outperforms state-of-the-art solvers in both speed and scalability.

The remainder of this paper is organized as follows. Section 2 introduces the HPR method with semi-proximal terms for solving the restricted Wolfe dual of CCQP. Section 3 details the implementation of HPR-QP, including its adaptive restart strategy and penalty parameter update scheme. In Section 4, we present extensive numerical results across various CCQP benchmark data sets. Finally, Section 5 concludes the paper with a summary and future directions.

Notation. Let $\mathbb{R}^n$ denote the n-dimensional real Euclidean space, equipped with the standard inner product $\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$. The infinity norm is denoted by $\|\cdot\|_\infty$, and the nonnegative orthant is denoted by $\mathbb{R}^n_+$. For a linear operator $A:\mathbb{R}^n\to\mathbb{R}^m$, we denote its adjoint by A^*, and its spectral norm by $\|A\|:=\sqrt{\lambda_1(AA^*)}$, where $\lambda_1(\cdot)$ denotes the largest eigenvalue of a self-adjoint linear operator. Furthermore, for any self-adjoint positive semidefinite linear operator $\mathcal{M}:\mathbb{R}^n\to\mathbb{R}^n$, we define the semi-norm $\|x\|_{\mathcal{M}}:=\sqrt{\langle x,\mathcal{M}x\rangle}$ for any $x\in\mathbb{R}^n$. Given a convex function $f:\mathbb{R}^n\to(-\infty,+\infty]$, we denote its effective domain by $\mathrm{dom}(f):=\{x\in\mathbb{R}^n\mid f(x)<+\infty\}$, its subdifferential by $\partial f(\cdot)$, its convex conjugate by $f^*(z):=\sup_{x\in\mathbb{R}^n}\{\langle x,z\rangle-f(x)\}$, and its proximal mapping by $\mathrm{Prox}_f(x):=\arg\min_{z\in\mathbb{R}^n}\left\{f(z)+\frac{1}{2}\|z-x\|^2\right\}$, respectively. Moreover, let $C\subseteq\mathbb{R}^n$ be a closed convex set. The indicator function over C, denoted by $\delta_C(\cdot)$, is defined as $\delta_C(x)=0$ if $x\in C$ and $\delta_C(x)=+\infty$ if $x\notin C$. The Euclidean distance from a point $x\in\mathbb{R}^n$ to C is $\mathrm{dist}(x,C):=\inf_{z\in C}\|z-x\|$, and the Euclidean projection onto C is $\Pi_C(x):=\arg\min_{z\in C}\|x-z\|$.

2 A Dual HPR Method for Solving CCQP

In this section, we present a general HPR framework with semi-proximal terms for solving the restricted Wolfe dual problem (1.5). In particular, it includes a variant based on the sGS technique for simplifying the solution of the subproblems as a special instance.

2.1 An HPR Method with Semi-proximal Terms

According to [39, Corollary 28.3.1], a point $(y^*,w^*,z^*)\in\mathbb{R}^m\times\mathcal{W}\times\mathbb{R}^n$ is an optimal solution to problem (1.5) if and only if there exists $x^*\in\mathbb{R}^n$ such that the KKT system below is satisfied:

$$Qw^*-Qx^*=0,\quad Ax^*\in\partial\delta^*_{\mathcal{K}}(-y^*),\quad x^*\in\partial\phi^*(-z^*),\quad -Qw^*+A^*y^*+z^*-c=0. \tag{2.1}$$

Let $\sigma>0$ be a given penalty parameter. Define the augmented Lagrangian function $L_\sigma(y,w,z;x)$ associated with problem (1.5), for any $(y,w,z,x)\in\mathbb{R}^m\times\mathcal{W}\times\mathbb{R}^n\times\mathbb{R}^n$, as follows

$$L_\sigma(y,w,z;x)=\frac{1}{2}\langle w,Qw\rangle+\delta^*_{\mathcal{K}}(-y)+\phi^*(-z)+\langle x,-Qw+A^*y+z-c\rangle+\frac{\sigma}{2}\left\|-Qw+A^*y+z-c\right\|^2.$$

For notational convenience, we denote the tuple (y,w,z,x) by u, and define the space $\mathcal{U}:=\mathbb{R}^m\times\mathcal{W}\times\mathbb{R}^n\times\mathbb{R}^n$. The HPR method with semi-proximal terms, which corresponds to the accelerated preconditioned ADMM with $\alpha=2$ proposed in [43], is presented in Algorithm 1 for solving the restricted Wolfe dual problem (1.5).

Algorithm 1 An HPR method for solving the restricted Wolfe dual problem (1.5)

Input: Choose a self-adjoint positive semidefinite linear operator $\mathcal{T}_1$ on $\mathbb{R}^m\times\mathcal{W}$. Denote $u=(y,w,z,x)$ and $\bar{u}=(\bar{y},\bar{w},\bar{z},\bar{x})$. Let $u^0=(y^0,w^0,z^0,x^0)\in\mathcal{U}$, and set $\sigma>0$.
for $k=0,1,2,\ldots$ **do**
  Step 1. $\bar{z}^{k+1}=\arg\min\limits_{z\in\mathbb{R}^n}\left\{L_\sigma(y^k,w^k,z;x^k)\right\}$;
  Step 2. $\bar{x}^{k+1}=x^k+\sigma(-Qw^k+A^*y^k+\bar{z}^{k+1}-c)$;
  Step 3. $(\bar{y}^{k+1},\bar{w}^{k+1})=\arg\min\limits_{(y,w)\in\mathbb{R}^m\times\mathcal{W}}\left\{L_\sigma(y,w,\bar{z}^{k+1};\bar{x}^{k+1})+\frac{1}{2}\|(y,w)-(y^k,w^k)\|^2_{\mathcal{T}_1}\right\}$;
  Step 4. $\hat{u}^{k+1}=2\bar{u}^{k+1}-u^k$;
  Step 5. $u^{k+1}=\frac{1}{k+2}u^0+\frac{k+1}{k+2}\hat{u}^{k+1}$;
end for

To analyze the global convergence of the HPR method with semi-proximal terms presented in Algorithm 1, we define the linear operator $\mathcal{H}:\mathbb{R}^m\times\mathcal{W}\to\mathbb{R}^m\times\mathcal{W}$ as follows:

$$\mathcal{H}:=\sigma A_Q^*A_Q+\begin{pmatrix}\mathbf{0}&\mathbf{0}\\ \mathbf{0}&Q\end{pmatrix}=\begin{pmatrix}\sigma AA^*&-\sigma AQ\\ -\sigma QA^*&\sigma Q^2+Q\end{pmatrix}, \tag{2.2}$$

where $A_Q := [A^* - Q] \in \mathbb{R}^{n\times(m+n)}$. Furthermore, we define a self-adjoint linear operator $\mathcal{M} : \mathbb{R}^m \times \mathcal{W} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m \times \mathcal{W} \times \mathbb{R}^n \times \mathbb{R}^n$ as

$$\mathcal{M} = \begin{bmatrix} \sigma A_Q^* A_Q + \mathcal{T}_1 & \mathbf{0} & A_Q^* \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ A_Q & \mathbf{0} & \frac{1}{\sigma} I_n \end{bmatrix}, \tag{2.3}$$

where $I_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix. Now, we make the following assumptions:

Assumption 2.1. *There exists a vector $(y^*, w^*, z^*, x^*) \in \mathbb{R}^m \times \mathcal{W} \times \mathbb{R}^n \times \mathbb{R}^n$ satisfying the KKT system (2.1).*

Assumption 2.2. *The operator $\mathcal{T}_1$ is a self-adjoint positive semidefinite linear operator such that $\mathcal{T}_1 + \mathcal{H}$ is positive definite on $\mathbb{R}^m \times \mathcal{W}$.*

Under Assumption 2.1, solving problems (1.1) and (1.5) is equivalent to finding a $u^* \in \mathcal{U}$ such that $\mathbf{0} \in \mathcal{T}u^*$, where the maximal monotone operator $\mathcal{T}$ is defined by

$$\mathcal{T}u = \begin{pmatrix} -\partial\delta_{\mathcal{K}}^*(-y) + Ax \\ Qw - Qx \\ -\partial\phi^*(-z) + x \\ c - A^*y + Qw - z \end{pmatrix} \quad \forall u = (y, w, z, x) \in \mathbb{R}^m \times \mathcal{W} \times \mathbb{R}^n \times \mathbb{R}^n. \tag{2.4}$$

Moreover, under Assumption 2.2, each subproblem in Algorithm 1 admits a unique solution. Based on Corollary 3.5 in [43], we establish the following global convergence result for the HPR method.

Proposition 2.1. *Suppose that Assumptions 2.1 and 2.2 hold. Then the sequence $\{\bar{u}^k\} = \{(\bar{y}^k, \bar{w}^k, \bar{z}^k, \bar{x}^k)\}$ generated by the HPR method with semi-proximal terms in Algorithm 1 converges to the point $u^* = (y^*, w^*, z^*, x^*)$, where (y^*, w^*, z^*) solves problem (1.5) and x^* solves problem (1.1).*

To further analyze the complexity of the HPR method with semi-proximal terms in terms of the KKT residual and the objective error, we consider the residual mapping associated with the KKT system (2.1), as introduced in [19]:

$$\mathcal{R}(u) = \begin{pmatrix} Ax - \Pi_{\mathcal{K}}(Ax - y) \\ Qw - Qx \\ x - \mathrm{Prox}_{\phi}(x - z) \\ c - A^*y + Qw - z \end{pmatrix} \quad \forall\, u = (y, w, z, x) \in \mathbb{R}^m \times \mathcal{W} \times \mathbb{R}^n \times \mathbb{R}^n. \tag{2.5}$$

In addition, let $\{(\bar{y}^k, \bar{w}^k, \bar{z}^k)\}$ be the sequence generated by Algorithm 1. We define the objective error as

$$\begin{aligned} h(\bar{y}^{k+1}, \bar{w}^{k+1}, \bar{z}^{k+1}) := {} & \frac{1}{2}\langle \bar{w}^{k+1}, Q\bar{w}^{k+1}\rangle + \delta_{\mathcal{K}}^*(-\bar{y}^{k+1}) + \phi^*(-\bar{z}^{k+1}) \\ & - \left(\frac{1}{2}\langle w^*, Qw^*\rangle + \delta_{\mathcal{K}}^*(-y^*) + \phi^*(-z^*)\right) \quad \forall\, k \geq 0, \end{aligned}$$

where (y^*, w^*, z^*) is a solution to the dual problem (1.5). Based on the iteration complexity results in Proposition 2.9, Theorem 3.7, and Remark 3.8 of [43], we obtain the following iteration complexity bounds for the HPR method with semi-proximal terms.

Proposition 2.2. *Suppose that Assumptions 2.1 and 2.2 hold. Let $\{\bar{u}^k\} = \{(\bar{y}^k, \bar{w}^k, \bar{z}^k, \bar{x}^k)\}$ and $\{u^k\} = \{(y^k, w^k, z^k, x^k)\}$ be the sequences generated by the HPR method with semi-proximal terms in Algorithm 1, and let $u^* = (y^*, w^*, z^*, x^*)$ be a solution to the KKT system (2.1). Define $R_0 := \|u^0 - u^*\|_{\mathcal{M}}$. Then, for all $k \geq 0$, the following complexity bounds hold:*

$$\|\bar{u}^{k+1} - u^k\|_{\mathcal{M}} \leq \frac{R_0}{k+1},$$

$$\|\mathcal{R}(\bar{u}^{k+1})\| \le \left(\frac{\sigma\|A_Q^*\|+1}{\sqrt{\sigma}} + \|\sqrt{\mathcal{T}_1}\|\right)\frac{R_0}{k+1},$$

and

$$-\frac{1}{\sqrt{\sigma}}\|x^*\|\cdot\frac{R_0}{k+1} \;\le\; h(\bar{y}^{k+1},\bar{w}^{k+1},\bar{z}^{k+1}) \;\le\; \left(3R_0+\frac{1}{\sqrt{\sigma}}\|x^*\|\right)\frac{R_0}{k+1}.$$

2.2 A Dual HPR Method Incorporating the sGS Technique

Note that the main computational bottleneck in Algorithm 1 lies in solving the subproblem with respect to the variables (y,w). To alleviate this difficulty, we apply the sGS technique [23, 26] to decouple the variables y and w. Specifically, we define the self-adjoint positive semidefinite linear operator $\mathcal{S}:\mathbb{R}^m\times\mathcal{W}\to\mathbb{R}^m\times\mathcal{W}$ as

$$\mathcal{S} = \sigma\begin{pmatrix}\mathcal{S}_y & \mathbf{0}\\ \mathbf{0} & \mathcal{S}_w\end{pmatrix}, \tag{2.6}$$

where $\mathcal{S}_y$ and $\mathcal{S}_w$ are self-adjoint positive semidefinite linear operators such that $\mathcal{S}_y+AA^*$ is positive definite on $\mathbb{R}^m$. Then the sGS operator $\widehat{\mathcal{S}}_{\rm sGS}:\mathbb{R}^m\times\mathcal{W}\to\mathbb{R}^m\times\mathcal{W}$ is defined as

$$\widehat{\mathcal{S}}_{\rm sGS} = \begin{pmatrix}\mathbf{0} & -\sigma AQ\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\frac{1}{\sigma}(AA^*+\mathcal{S}_y)^{-1} & \mathbf{0}\\ \mathbf{0} & (\sigma Q^2+Q+\sigma\mathcal{S}_w)^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ -\sigma QA^* & \mathbf{0}\end{pmatrix} = \begin{pmatrix}\widehat{\mathcal{S}}_{\rm sGS1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}, \tag{2.7}$$

where $\widehat{\mathcal{S}}_{\rm sGS1} := \sigma^2 AQ(\sigma Q^2+Q+\sigma\mathcal{S}_w)^{-1}QA^*$. According to [26, Theorem 1], the following proposition shows that incorporating the sGS operator enables an efficient block-wise update of $(\bar{y}^{k+1},\bar{w}^{k+1})$ for all $k\ge 0$.

Proposition 2.3. *Let $\mathcal{S}_y$ and $\mathcal{S}_w$ be two self-adjoint positive semidefinite linear operators on $\mathbb{R}^m$ and $\mathcal{W}$, respectively, such that $\mathcal{S}_y+AA^*$ is positive definite. Let $\mathcal{T}_1=\mathcal{S}+\widehat{\mathcal{S}}_{\rm sGS}$, where $\mathcal{S}$ and $\widehat{\mathcal{S}}_{\rm sGS}$ are given in (2.6) and (2.7). Then for any $k\ge 0$, Step 3 of Algorithm 1,*

$$(\bar{y}^{k+1},\bar{w}^{k+1}) = \mathop{\arg\min}_{(y,w)\in\mathbb{R}^m\times\mathcal{W}}\left\{L_\sigma(y,w,\bar{z}^{k+1};\bar{x}^{k+1})+\frac{1}{2}\|(y,w)-(y^k,w^k)\|^2_{\mathcal{T}_1}\right\}$$

is equivalent to the following updates:

$$\begin{cases}\bar{w}^{k+\frac{1}{2}} = \mathop{\arg\min}_{w\in\mathcal{W}}\left\{L_\sigma(y^k,w,\bar{z}^{k+1};\bar{x}^{k+1})+\frac{\sigma}{2}\|w-w^k\|^2_{\mathcal{S}_w}\right\},\\ \bar{y}^{k+1} = \mathop{\arg\min}_{y\in\mathbb{R}^m}\left\{L_\sigma(y,\bar{w}^{k+\frac{1}{2}},\bar{z}^{k+1};\bar{x}^{k+1})+\frac{\sigma}{2}\|y-y^k\|^2_{\mathcal{S}_y}\right\},\\ \bar{w}^{k+1} = \mathop{\arg\min}_{w\in\mathcal{W}}\left\{L_\sigma(\bar{y}^{k+1},w,\bar{z}^{k+1};\bar{x}^{k+1})+\frac{\sigma}{2}\|w-w^k\|^2_{\mathcal{S}_w}\right\}.\end{cases}$$

Moreover, $\mathcal{T}_1+\mathcal{H}$ is positive definite on $\mathbb{R}^m\times\mathcal{W}$.

An HPR method incorporating the sGS technique is presented in Algorithm 2:

Algorithm 2 A dual HPR method for solving the restricted-Wolfe dual problem (1.5)

Input: Let $\mathcal{S}_y$ and $\mathcal{S}_w$ be two self-adjoint positive semidefinite linear operators on $\mathbb{R}^m$ and $\mathcal{W}$, respectively, such that $\mathcal{S}_y + AA^*$ is positive definite. Denote $u = (y, w, z, x)$ and $\bar{u} = (\bar{y}, \bar{w}, \bar{z}, \bar{x})$. Let $u^0 = (y^0, w^0, z^0, x^0) \in \mathcal{U}$, and set $\sigma > 0$.

for $k = 0, 1, 2, \ldots$ **do**

Step 1. $\bar{z}^{k+1} = \arg\min_{z \in \mathbb{R}^n} \left\{ L_\sigma(y^k, w^k, z; x^k) \right\}$;

Step 2. $\bar{x}^{k+1} = x^k + \sigma(-Qw^k + A^* y^k + \bar{z}^{k+1} - c)$;

Step 3-1. $\bar{w}^{k+\frac{1}{2}} = \arg\min_{w \in \mathcal{W}} \left\{ L_\sigma(y^k, w, \bar{z}^{k+1}; \bar{x}^{k+1}) + \frac{\sigma}{2} \|w - w^k\|^2_{\mathcal{S}_w} \right\}$;

Step 3-2. $\bar{y}^{k+1} = \arg\min_{y \in \mathbb{R}^m} \left\{ L_\sigma(y, \bar{w}^{k+\frac{1}{2}}, \bar{z}^{k+1}; \bar{x}^{k+1}) + \frac{\sigma}{2} \|y - y^k\|^2_{\mathcal{S}_y} \right\}$;

Step 3-3. $\bar{w}^{k+1} = \arg\min_{w \in \mathcal{W}} \left\{ L_\sigma(\bar{y}^{k+1}, w, \bar{z}^{k+1}; \bar{x}^{k+1}) + \frac{\sigma}{2} \|w - w^k\|^2_{\mathcal{S}_w} \right\}$;

Step 4. $\hat{u}^{k+1} = 2\bar{u}^{k+1} - u^k$;

Step 5. $u^{k+1} = \frac{1}{k+2} u^0 + \frac{k+1}{k+2} \hat{u}^{k+1}$;

end for

Remark 2.1. *The proposed HPR method, as outlined in Algorithm 2, which incorporates the sGS technique, provides a unified and flexible framework for solving general CCQP problems of the form (1.1). In particular, when $Q = \mathbf{0}$ and $\phi(\cdot) = \delta_C(\cdot)$, Algorithm 2 reduces exactly to the HPR method for LP introduced in [8]. Moreover, if the constraint $Ax \in \mathcal{K}$ is absent—as in the case of the Lasso problem [24]—then Algorithm 2 simplifies to the HPR method with the update cycle $z \to x \to w$. These special cases highlight the flexibility and broad applicability of the dual HPR method in Algorithm 2.*

Remark 2.2. *In Algorithm 2, the updates for $\bar{w}^{k+\frac{1}{2}}$ and $\bar{w}^{k+1}$ for $k \geq 0$ are restricted to the subspace $\mathcal{W} = \mathrm{Range}(Q)$. Although it may seem more straightforward to update $\bar{w}^{k+\frac{1}{2}}$ and $\bar{w}^{k+1}$ in the full space $\mathbb{R}^n$, doing so—particularly under a linearized ADMM framework—necessitates a proximal operator with a larger spectral norm, such as $\mathcal{S}_w = \lambda_1(Q^2 + Q/\sigma) I_n - (Q^2 + Q/\sigma)$, to ensure convergence. A proximal operator with a large spectral norm typically results in slower convergence. By contrast, restricting the update to $\mathcal{W}$ allows HPR-QP to employ a proximal operator with a smaller spectral norm, namely $\mathcal{S}_w = Q(\lambda_1(Q) I_n - Q)$, which accelerates convergence while preserving theoretical guarantees.*

2.3 An Easy-to-Implement Dual HPR Method

While directly computing $\bar{w}^{k+\frac{1}{2}}$ and $\bar{w}^{k+1}$ for $k \geq 0$ within the subspace $\mathrm{Range}(Q)$ may appear computationally intensive, we show in this subsection that these updates can be performed efficiently without requiring explicit projection onto $\mathrm{Range}(Q)$. For small-scale problems or when Q has a favorable structure, one may set $\mathcal{S}_w = 0$ and solve the subproblems using direct solvers or preconditioned conjugate gradient methods [27]. Here, for solving large-scale general CCQP problems, we employ the proximal operator

$$\mathcal{S}_w = Q(\lambda_Q I_n - Q), \tag{2.8}$$

where $\lambda_Q > 0$ is a constant satisfying $\lambda_Q \geq \lambda_1(Q)$. Then, for any $k \geq 0$, the updates of $\bar{w}^{k+\frac{1}{2}}$ and $\bar{w}^{k+1}$ are given by

$$\begin{cases} Q\bar{w}^{k+\frac{1}{2}} = \dfrac{1}{1+\sigma\lambda_Q} Q\left(\sigma\lambda_Q w^k + \bar{x}^{k+1} + \sigma(-Qw^k + A^* y^k + \bar{z}^{k+1} - c)\right), \quad \bar{w}^{k+\frac{1}{2}} \in \mathcal{W}, \\ Q\bar{w}^{k+1} = \dfrac{1}{1+\sigma\lambda_Q} Q\left(\sigma\lambda_Q w^k + \bar{x}^{k+1} + \sigma(-Qw^k + A^* \bar{y}^{k+1} + \bar{z}^{k+1} - c)\right), \quad \bar{w}^{k+1} \in \mathcal{W}. \end{cases}$$

The following proposition—motivated by [25, Proposition 4.1]—demonstrates that this choice of $\mathcal{S}_w$ enables efficient computation of the updates by constructing a shadow sequence, avoiding explicit projection onto $\mathrm{Range}(Q)$.

Proposition 2.4. *Let $\mathcal{W} = \mathrm{Range}(Q)$ and $R \in \mathbb{R}^n$. To solve*

$$Q\bar{w}^+ = \frac{1}{1+\sigma\lambda_Q} QR, \quad \bar{w}^+ \in \mathcal{W}, \tag{2.9}$$

it suffices to set

$$\bar{w}_Q^+ := \frac{1}{1+\sigma\lambda_Q} R. \tag{2.10}$$

Then $\bar{w}^+ = \Pi_{\mathcal{W}}(\bar{w}_Q^+)$ solves (2.9), and $Q\bar{w}^+ = Q\bar{w}_Q^+$, $\langle \bar{w}^+, Q\bar{w}^+ \rangle = \langle \bar{w}_Q^+, Q\bar{w}_Q^+ \rangle$.

Based on Proposition 2.4, an easy-to-implement dual HPR method for solving the large-scale restricted-Wolfe dual problem (1.5) is presented in Algorithm 3.

Algorithm 3 An easy-to-implement dual HPR method for the restricted-Wolfe dual problem (1.5)

Input: Let $\mathcal{S}_w$ be defined as in (2.8), and let $\mathcal{S}_y$ be a self-adjoint positive semidefinite linear operator on $\mathbb{R}^m$ such that $\mathcal{S}_y + AA^*$ is positive definite. Denote $u_Q = (y, w_Q, z, x)$ and $\bar{u}_Q = (\bar{y}, \bar{w}_Q, \bar{z}, \bar{x})$. Let $u_Q^0 = (y^0, w_Q^0, z^0, x^0) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$, and set $\sigma > 0$.

for $k = 0, 1, 2, \ldots$ **do**

Step 1. $\bar{z}^{k+1} = \arg\min_{z \in \mathbb{R}^n} \left\{ L_\sigma(y^k, w_Q^k, z; x^k) \right\}$;

Step 2. $\bar{x}^{k+1} = x^k + \sigma(-Qw_Q^k + A^* y^k + \bar{z}^{k+1} - c)$;

Step 3-1. $\bar{w}_Q^{k+\frac{1}{2}} = \frac{1}{1+\sigma\lambda_Q} \left(\sigma\lambda_Q w_Q^k + \bar{x}^{k+1} + \sigma(-Qw_Q^k + A^* y^k + \bar{z}^{k+1} - c) \right)$;

Step 3-2. $\bar{y}^{k+1} = \arg\min_{y \in \mathbb{R}^m} \left\{ L_\sigma(y, \bar{w}_Q^{k+\frac{1}{2}}, \bar{z}^{k+1}; \bar{x}^{k+1}) + \frac{\sigma}{2} \|y - y^k\|_{\mathcal{S}_y}^2 \right\}$;

Step 3-3. $\bar{w}_Q^{k+1} = \frac{1}{1+\sigma\lambda_Q} \left(\sigma\lambda_Q w_Q^k + \bar{x}^{k+1} + \sigma(-Qw_Q^k + A^* \bar{y}^{k+1} + \bar{z}^{k+1} - c) \right)$;

Step 4. $\hat{u}_Q^{k+1} = 2\bar{u}_Q^{k+1} - u_Q^k$;

Step 5. $u_Q^{k+1} = \frac{1}{k+2} u_Q^0 + \frac{k+1}{k+2} \hat{u}_Q^{k+1}$;

end for

Theorem 2.1. *Suppose Assumption 2.1 holds. Then the sequence $\{(\bar{y}^k, Q\bar{w}_Q^k, \bar{z}^k, \bar{x}^k)\}$ generated by Algorithm 3 is equivalent to the sequence $\{(\bar{y}^k, Q\bar{w}^k, \bar{z}^k, \bar{x}^k)\}$ produced by Algorithm 2 starting from the same initial point. Moreover, both sequences converge to the point (y^*, Qw^*, z^*, x^*), where (y^*, w^*, z^*) solves problem (1.5) and x^* solves problem (1.1).*

Proof. According to the optimality conditions of the subproblems in Algorithm 2, for any $k \geq 0$, we have:

$$\begin{cases} \mathbf{0} \in -\partial\phi^*(-\bar{z}^{k+1}) + x^k + \sigma(-Qw^k + A^* y^k + \bar{z}^{k+1} - c), \\ \bar{x}^{k+1} = x^k + \sigma(-Qw^k + A^* y^k + \bar{z}^{k+1} - c), \\ Q\bar{w}^{k+\frac{1}{2}} = \frac{1}{1+\sigma\lambda_Q} Q \left(\sigma\lambda_Q w^k + \bar{x}^{k+1} + \sigma(-Qw^k + A^* y^k + \bar{z}^{k+1} - c) \right), \quad \bar{w}^{k+\frac{1}{2}} \in \mathcal{W}, \\ \mathbf{0} \in -\partial\delta_{\mathcal{K}}^*(-\bar{y}^{k+1}) + A\bar{x}^{k+1} + \sigma A(-Q\bar{w}^{k+\frac{1}{2}} + A^* \bar{y}^{k+1} + \bar{z}^{k+1} - c) + \sigma\mathcal{S}_y(\bar{y}^{k+1} - y^k), \\ Q\bar{w}^{k+1} = \frac{1}{1+\sigma\lambda_Q} Q \left(\sigma\lambda_Q w^k + \bar{x}^{k+1} + \sigma(-Qw^k + A^* \bar{y}^{k+1} + \bar{z}^{k+1} - c) \right), \quad \bar{w}^{k+1} \in \mathcal{W}. \end{cases} \tag{2.11}$$

Based on (2.11) and Proposition 2.4, we can derive that the sequence $\{(\bar{y}^k, Q\bar{w}_Q^k, \bar{z}^k, \bar{x}^k)\}$ generated by Algorithm 3 is equivalent to $\{(\bar{y}^k, Q\bar{w}^k, \bar{z}^k, \bar{x}^k)\}$ from Algorithm 2, under the same initialization.

Furthermore, Proposition 2.3 establishes that Algorithm 2 is a special case of Algorithm 1 with $\mathcal{T}_1 = \mathcal{S} + \widehat{\mathcal{S}}_{\mathrm{sGS}}$. Thus, the convergence result in Proposition 2.1 applies, and the sequence $\{(\bar{y}^k, \bar{w}^k, \bar{z}^k, \bar{x}^k)\}$ from Algorithm 2 converges to the optimal solution of the primal-dual pair (1.1) and (1.5). This completes the proof.

□

By substituting $\mathcal{T}_1 = \mathcal{S} + \widehat{\mathcal{S}}_{\rm sGS}$, where $\mathcal{S}$ and $\widehat{\mathcal{S}}_{\rm sGS}$ are defined in (2.6) and (2.7), respectively, into the definition of $\mathcal{M}$ in (2.3), we obtain the explicit form of $\mathcal{M}$ used in the dual HPR method (Algorithm 3):

$$\mathcal{M} = \begin{bmatrix} \sigma A_Q^* A_Q + \mathcal{S} + \widehat{\mathcal{S}}_{\rm sGS} & \mathbf{0} & A_Q^* \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ A_Q & \mathbf{0} & \frac{1}{\sigma} I_n \end{bmatrix}. \tag{2.12}$$

Combining Propositions 2.2, 2.3, and Theorem 2.1, we derive the following complexity result for Algorithm 3:

Theorem 2.2. *Suppose Assumption 2.1 holds. Let* $\{\bar{u}_Q^k\} = \{(y^k, w_Q^k, z^k, x^k)\}$ *and* $\{u_Q^k\} = \{(y^k, w_Q^k, z^k, x^k)\}$ *be the sequences generated by Algorithm 3, and let* $u^* = (y^*, w^*, z^*, x^*)$ *be a solution to the KKT system (2.1). Then, for all* $k \geq 0$*, the following complexity bounds hold with* $R_0 = \|u_Q^0 - u^*\|_{\mathcal{M}}$*:*

$$\begin{aligned} \|\bar{u}_Q^{k+1} - u_Q^k\|_{\mathcal{M}} &\leq \frac{R_0}{k+1}, \\ \|\mathcal{R}(\bar{u}_Q^{k+1})\| &\leq \left(\frac{\sigma \|A_Q^*\| + 1}{\sqrt{\sigma}} + \|\sqrt{\mathcal{S} + \widehat{\mathcal{S}}_{\rm sGS}}\| \right) \frac{R_0}{k+1}, \\ -\frac{\|x^*\|}{\sqrt{\sigma}} \cdot \frac{R_0}{k+1} \leq h(\bar{y}^{k+1}, \bar{w}_Q^{k+1}, \bar{z}^{k+1}) &\leq \left(3R_0 + \frac{\|x^*\|}{\sqrt{\sigma}} \right) \frac{R_0}{k+1}. \end{aligned}$$

3 HPR-QP: A Dual HPR Method for CCQP

In this section, we present HPR-QP, as outlined in Algorithm 4, for solving problem (1.5), incorporating an adaptive restart strategy and a dynamic update rule for σ.

3.1 Efficient Solution of Subproblems

We first detail the update formulas for each subproblem in HPR-QP (Steps 6–10). Specifically, for any $r \geq 0$ and $t \geq 0$, the update of $\bar{z}^{r,t+1}$ is given by

$$\bar{z}^{r,t+1} = \frac{1}{\sigma_r} \left(\mathrm{Prox}_{\sigma_r \phi}(r_z^{r,t}) - r_z^{r,t} \right), \tag{3.1}$$

where $r_z^{r,t} = x^{r,t} + \sigma_r(-Qw_Q^{r,t} + A^* y^{r,t} - c)$. The corresponding update for $\bar{x}^{r,t+1}$ becomes

$$\bar{x}^{r,t+1} = x^{r,t} + \sigma_r(-Qw_Q^{r,t} + A^* y^{r,t} + \bar{z}^{r,t+1} - c) = \mathrm{Prox}_{\sigma_r \phi}(r_z^{r,t}). \tag{3.2}$$

Furthermore, the updates for $\bar{w}_Q^{r,t+\frac{1}{2}}$ and $\bar{w}_Q^{r,t+1}$ can be simplified as follows

$$\begin{cases} \bar{w}_Q^{r,t+\frac{1}{2}} = \dfrac{1}{1+\sigma_r\lambda_Q} \left(\sigma_r \lambda_Q w_Q^{r,t} + \bar{x}^{r,t+1} + \sigma_r(-Qw_Q^{r,t} + A^* y^{r,t} + \bar{z}^{r,t+1} - c) \right) \\ \qquad\qquad = \dfrac{1}{1+\sigma_r\lambda_Q} \left(\sigma_r \lambda_Q w_Q^{r,t} + 2\bar{x}^{r,t+1} - x^{r,t} \right) = \dfrac{1}{1+\sigma_r\lambda_Q} \left(\sigma_r \lambda_Q w_Q^{r,t} + \hat{x}^{r,t+1} \right), \\ \bar{w}_Q^{r,t+1} = \dfrac{1}{1+\sigma_r\lambda_Q} \left(\sigma_r \lambda_Q w_Q^{r,t} + \bar{x}^{r,t+1} + \sigma_r(-Qw_Q^{r,t} + A^* \bar{y}^{r,t+1} + \bar{z}^{r,t+1} - c) \right) \\ \qquad\qquad = \left(\bar{w}_Q^{r,t+\frac{1}{2}} + \dfrac{\sigma_r}{1+\sigma_r\lambda_Q} A^*(\bar{y}^{r,t+1} - y^{r,t}) \right). \end{cases} \tag{3.3}$$

To simplify the solution of the subproblem with respect to y, we choose the proximal operator

$$\mathcal{S}_y = \lambda_A I_m - AA^*, \tag{3.4}$$

Algorithm 4 HPR-QP: A dual HPR method for the CCQP (1.5)

Input: Let $\mathcal{S}_w$ be defined as in (2.8), and let $\mathcal{S}_y$ be a self-adjoint, positive semidefinite operator on $\mathbb{R}^m$ such that $\mathcal{S}_y + AA^*$ is positive definite. Let $u_Q = (y, w_Q, z, x)$, $\bar{u}_Q = (\bar{y}, \bar{w}_Q, \bar{z}, \bar{x})$, and initial point $u_Q^{0,0} = (y^{0,0}, w_Q^{0,0}, z^{0,0}, x^{0,0}) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$.
Initialization: Set outer loop counter $r = 0$, total iteration counter $k = 0$, and initial penalty parameter $\sigma_0 > 0$.
repeat
  Initialize inner loop: set inner counter $t = 0$;
  repeat
    $\bar{z}^{r,t+1} = \arg\min_{z \in \mathbb{R}^n} L_{\sigma_r}(y^{r,t}, w_Q^{r,t}, z; x^{r,t})$;
    $\bar{x}^{r,t+1} = x^{r,t} + \sigma_r(-Qw_Q^{r,t} + A^* y^{r,t} + \bar{z}^{r,t+1} - c)$;
    $\bar{w}_Q^{r,t+\frac{1}{2}} = \frac{1}{1+\sigma_r\lambda_Q}\left(\sigma_r\lambda_Q w_Q^{r,t} + \bar{x}^{r,t+1} + \sigma_r(-Qw_Q^{r,t} + A^* y^{r,t} + \bar{z}^{r,t+1} - c)\right)$;
    $\bar{y}^{r,t+1} = \arg\min_{y \in \mathbb{R}^m}\left\{L_{\sigma_r}(y, \bar{w}_Q^{r,t+\frac{1}{2}}, \bar{z}^{r,t+1}; \bar{x}^{r,t+1}) + \frac{\sigma_r}{2}\|y - y^{r,t}\|_{\mathcal{S}_y}^2\right\}$;
    $\bar{w}_Q^{r,t+1} = \frac{1}{1+\sigma_r\lambda_Q}\left(\sigma_r\lambda_Q w_Q^{r,t} + \bar{x}^{r,t+1} + \sigma_r(-Qw_Q^{r,t} + A^* \bar{y}^{r,t+1} + \bar{z}^{r,t+1} - c)\right)$;
    $\hat{u}_Q^{r,t+1} = 2\bar{u}_Q^{r,t+1} - u_Q^{r,t}$;
    $u_Q^{r,t+1} = \frac{1}{k+2}u_Q^{r,0} + \frac{k+1}{k+2}\hat{u}_Q^{r,t+1}$;
    $t \leftarrow t+1$, $k \leftarrow k+1$;
  until restart or termination criteria are met;
  Restart: Set $\tau_r = t$, $u_Q^{r+1,0} = \bar{u}_Q^{r,\tau_r}$;
  $\sigma_{r+1} = \textbf{SigmaUpdate}(\bar{u}_Q^{r,\tau_r}, u_Q^{r,0}, \mathcal{S}_y, \mathcal{S}_w, A, Q)$;
  $r \leftarrow r+1$;
until termination criteria are met;
Output: $\{\bar{u}^{r,t}\}$.

where $\lambda_A > 0$ is a constant such that $\lambda_A \geq \lambda_1(AA^*)$. The updates for $\bar{y}^{r,t+1}$ are then given by

$$\bar{y}^{r,t+1} = \frac{1}{\sigma_r\lambda_A}\left(\Pi_{\mathcal{K}}(r_y^{r,t}) - r_y^{r,t}\right), \tag{3.5}$$

where

$$\begin{aligned} r_y^{r,t} &= A\left(\bar{x}^{r,t+1} + \sigma_r(-Q\bar{w}_Q^{r,t+\frac{1}{2}} + A^* y^{r,t} + \bar{z}^{r,t+1} - c)\right) - \sigma_r\lambda_A y^{r,t} \\ &= A(\hat{x}^{r,t+1} + \sigma_r(Qw_Q^{r,t} - Q\bar{w}_Q^{r,t+\frac{1}{2}})) - \sigma_r\lambda_A y^{r,t}. \end{aligned}$$

Moreover, by combining the update formulas in (3.2), (3.3), and (3.5), we find that it is not necessary to compute $\bar{z}^{r,t+1}$ at every iteration. Instead, $\bar{z}^{r,t+1}$ needs to be evaluated via (3.1) only when checking the stopping criteria. This further improves computational efficiency without affecting the correctness of the method.

Finally, to fully leverage the parallel computing capabilities of GPUs, we implement custom CUDA kernels for the updates in (3.1), (3.2), (3.3), and (3.5). For matrix-vector multiplications, we selectively use either custom kernel implementations or the `cusparseSpMV()` routine from the cuSPARSE library, which applies the `CUSPARSE_SPMV_CSR_ALG2` algorithm. Our custom implementations reduce the number of kernel launches, which is beneficial for performance, especially when minimizing launch overhead. However, they may be less efficient than cuSPARSE when handling dense or moderately sparse matrices. Therefore, when A and Q are not highly sparse, we prefer `cusparseSpMV()` to ensure optimal performance.

3.2 An Adaptive Restart Strategy

Inspired by the restart strategy proposed in PDLP [2, 32, 34], HPR-LP [8] introduced an adaptive restart mechanism based on the $O(1/k)$ iteration complexity of the HPR method, which has demonstrated practical success on large-scale LP problems. Motivated by this, we extend this adaptive restart strategy to the CCQP (1.1) by defining a suitable merit function grounded in the complexity result established in Theorem 2.2. Specifically, we define the following idealized merit function:

$$R_{r,t} := \|u_Q^{r,t} - u^*\|_{\mathcal{M}} \quad \forall r \geq 0,\ t \geq 0,$$

where u^* denotes any solution to the KKT system (2.1). Note that $R_{r,0}$ corresponds to the upper bound given by the complexity result in Theorem 2.2 at the start of the r-th outer iteration. Since u^* is unknown in practice, we approximate $R_{r,t}$ using the computable surrogate:

$$\widetilde{R}_{r,t} := \|u_Q^{r,t} - \bar{u}_Q^{r,t+1}\|_{\mathcal{M}}.$$

Based on this surrogate merit function, we introduce the following adaptive restart criteria for the HPR-QP method:

1. Sufficient decay:
$$\widetilde{R}_{r,t+1} \leq \alpha_1 \widetilde{R}_{r,0}; \tag{3.6}$$

2. Insufficient local progress despite overall decay:
$$\widetilde{R}_{r,t+1} \leq \alpha_2 \widetilde{R}_{r,0} \quad \text{and} \quad \widetilde{R}_{r,t+1} > \widetilde{R}_{r,t}; \tag{3.7}$$

3. Excessively long inner loop:
$$t \geq \alpha_3 k; \tag{3.8}$$

where $\alpha_1 \in (0, \alpha_2)$, $\alpha_2 \in (0, 1)$, and $\alpha_3 \in (0, 1)$ are user-defined parameters. Whenever any of the above conditions is satisfied, we terminate the current inner loop and begin a new outer iteration by setting $u^{r+1,0} = \bar{u}^{r,\tau_r}$ and updating the penalty parameter σ_{r+1} accordingly.

3.3 An Update Strategy for σ

Now, we describe the update rule for the penalty parameter σ in HPR-QP. This design is also motivated by the complexity results of the HPR method with semi-proximal terms, as established in Theorem 2.2. Specifically, at the beginning of the $(r+1)$-th outer iteration, we determine σ_{r+1} by solving the following minimization problem:

$$\sigma_{r+1} = \arg\min_{\sigma>0} \|u_Q^{r+1,0} - u^*\|_{\mathcal{M}}^2, \tag{3.9}$$

where u^* denotes a solution to the KKT system (2.1), and the metric $\|u_Q^{r+1,0} - u^*\|_{\mathcal{M}}$ corresponds to the upper bound derived in Theorem 2.2. The rationale behind this formulation is that minimizing this bound is expected to yield a smaller quantity $\|u_Q^{r+1,t} - \bar{u}_Q^{r+1,t+1}\|_{\mathcal{M}}$ for all $t \geq 0$, which, in turn, reduces the KKT residual $\|\mathcal{R}(\bar{u}_Q^{r+1,t+1})\|$. This insight guides the dynamic adjustment of σ to enhance the practical performance of the method.

Substituting (2.12) and (3.4) into (3.9), the update rule for σ_{r+1} reduces to solving the following one-dimensional minimization problem:

$$\sigma_{r+1} = \arg\min_{\sigma>0} \left\{ f(\sigma) := \theta_1 \sigma + \frac{\theta_2}{\sigma} + \frac{\sigma^2 \theta_3}{1 + \lambda_Q \sigma} \right\}, \tag{3.10}$$

where the coefficients $\theta_1, \theta_2, \theta_3$ are given by

$$\theta_1 = \lambda_A \|y^{r+1,0} - y^*\|^2 + \lambda_Q \|w_Q^{r+1,0} - w^*\|_Q^2 - 2\langle Q(w_Q^{r+1,0} - w^*), A^*(y^{r+1,0} - y^*)\rangle,$$
$$\theta_2 = \|x^{r+1,0} - x^*\|^2, \quad \theta_3 = \|A^* y^{r+1,0} - A^* y^*\|_Q^2.$$

When coefficients $\theta_i > 0$, $i = 2, 3$, the function $f(\sigma)$ is strictly convex on $\sigma > 0$, since its second derivative satisfies

$$f''(\sigma) = \frac{2\theta_2}{\sigma^3} + \frac{2\theta_3}{(1+\lambda_Q\sigma)^3} > 0 \quad \text{for all } \sigma > 0.$$

This guarantees both the existence and uniqueness of the optimal solution to problem (3.10). This scalar optimization problem can be efficiently solved using the golden section search method [14]. Moreover, since the exact values of θ_i involve the unknown solution u^*, we approximate them in practice as follows:

$$\begin{aligned} &\widetilde{\theta}_1 = \lambda_A\|\bar{y}^{r,\tau_r} - y^{r,0}\|^2 + \lambda_Q\|\bar{w}_Q^{r,\tau_r} - w_Q^{r,0}\|_Q^2 - 2\langle Q(\bar{w}_Q^{r,\tau_r} - w_Q^{r,0}), A^*(\bar{y}^{r,\tau_r} - y^{r,0})\rangle, \\ &\widetilde{\theta}_2 = \|\bar{x}^{r,\tau_r} - x^{r,0}\|^2, \quad \widetilde{\theta}_3 = \|A^*\bar{y}^{r,\tau_r} - A^*y^{r,0}\|_Q^2. \end{aligned} \tag{3.11}$$

Using these approximations, the updated penalty parameter σ is obtained by solving the following scalar optimization problem:

$$\sigma_{\text{new}} = \arg\min_{\sigma>0}\left\{ f(\sigma) = \widetilde{\theta}_1\sigma + \frac{\widetilde{\theta}_2}{\sigma} + \frac{\sigma^2\widetilde{\theta}_3}{1+\lambda_Q\sigma} \right\}. \tag{3.12}$$

To further stabilize the update and prevent the approximations from deviating significantly from the true values, we apply an exponential smoothing scheme. The complete update procedure for σ is summarized in Algorithm 5:

Algorithm 5 SigmaUpdate

1: **Input:** $\bar{u}_Q^{r,\tau_r}, u_Q^{r,0}, \mathcal{S}_y, \mathcal{S}_w, A, Q$.
2: Compute $\widetilde{\theta}_1, \widetilde{\theta}_2, \widetilde{\theta}_3$ as defined in (3.11);
3: Ensure numerical stability: $\widetilde{\theta}_i \leftarrow \max\big(\widetilde{\theta}_i, 10^{-12}\big)$, $i = 1, 2$;
4: Solve problem (3.12) using golden section search method to obtain σ_{new};
5: Compute smoothing factor: $\beta = \exp\left(-\widetilde{R}_{r,\tau_r-1}/\widetilde{R}_{0,\tau_0-1}\right)$;
6: **Output:** $\sigma_{r+1} = \exp\big(\beta\log(\sigma_{\text{new}}) + (1-\beta)\log(\sigma_r)\big)$.

4 Numerical Experiments

In this section, we evaluate the performance of the Julia implementation of HPR-QP on a GPU and compare it against several state-of-the-art CQP solvers, including PDQP [33], SCS [37, 36], CuClarabel [10], and Gurobi [17]. The details of the experimental setup are provided in Section 4.1. We report numerical results on a diverse set of benchmark problems: Section 4.2 presents results on the classical Maros-Mészáros data set [35]; Section 4.3 covers large-scale synthetic CQP instances from six problem classes introduced in [42]; Section 4.4 focuses on Lasso regression problems; and Section 4.5 reports results on large-scale CQP problems arising from relaxations of QAPs.

4.1 Experimental Setup

Solvers and Computing Environment. The HPR-QP solver is implemented in Julia [4] with GPU acceleration enabled via CUDA. For comparison, both PDQP[1] and CuClarabel[2] are also implemented in Julia with support for CUDA-based GPU execution. In contrast, SCS[3] is developed in C/C++ with a Julia interface and uses GPU acceleration through its indirect solver, where all matrix operations are performed on the GPU. Gurobi (version 12.0.2, academic license) is executed on the CPU. All solvers are benchmarked on a SuperServer SYS-420GP-TNR equipped with an NVIDIA A100-SXM4-80GB GPU, an Intel Xeon Platinum 8338C CPU @ 2.60 GHz, and 256 GB of RAM. The experiments are conducted on Ubuntu 24.04.2 LTS.

[1] https://github.com/jinwen-yang/PDQP.jl, downloaded in April 2025
[2] https://github.com/oxfordcontrol/Clarabel.jl/tree/CuClarabel, version 0.10.0
[3] https://github.com/jump-dev/SCS.jl, version 2.1.0

Preconditioning. Similar to PDQP [33] and HPR-LP [8], HPR-QP also employs a diagonal preconditioning strategy when Q is available in explicit matrix form. In this setting, we perform 10 iterations of Ruiz equilibration [40], followed by the diagonal preconditioning method of Pock and Chambolle [38] with parameter $\alpha = 1$. If Q is provided implicitly (i.e., as a matrix-free operator), preconditioning is not applied in the current implementation.

Parameter Setting. After preconditioning, we estimate λ_A and λ_Q using the power method [15]. HPR-QP is initialized at the origin, and the initial penalty parameter is set to $\sigma_0 = \|b\|/\|c\|$, when both $\|b\|$ and $\|c\|$ lie within the range $[10^{-16}, 10^{16}]$; here, $b := \max(|l|, |u|)$ is taken componentwise, treating any infinite entries in l or u as zero when computing the norm. Otherwise, we set $\sigma_0 = 1$ to ensure numerical stability. The adaptive restart mechanism follows the criteria in (3.6)–(3.8), with parameters $\alpha_1 = 0.2$, $\alpha_2 = 0.8$, and $\alpha_3 = 0.5$. If the residual ratio satisfies $\widetilde{R}_{r,\tau_r-1}/\widetilde{R}_{0,\tau_0-1} \leq 0.1$, then α_3 is tightened to 0.2. The penalty parameter σ is dynamically updated according to Algorithm 5. All other solvers are executed using their default parameter settings.

Termination and Time Limit. We terminate HPR-QP when the stopping criteria, similar to those in PDQP [33], are satisfied for a given tolerance $\varepsilon \in (0, \infty)$. Specifically, the solver stops when the following conditions hold:

$$\eta_{\text{gap}} = \frac{\left|\frac{1}{2}\langle x, Qx\rangle + \langle c, x\rangle + \phi(x) - \left(-\frac{1}{2}\langle x, Qx\rangle - \delta^*_{\mathcal{K}}(-y) - \phi^*(-z)\right)\right|}{1 + \max\left(\left|\frac{1}{2}\langle x, Qx\rangle + \langle c, x\rangle + \phi(x)\right|, \left|\frac{1}{2}\langle x, Qx\rangle + \delta^*_{\mathcal{K}}(-y) + \phi^*(-z)\right|\right)} \leq \varepsilon,$$

$$\eta_{\text{p}} = \frac{\|Ax - \Pi_{\mathcal{K}}(Ax)\|_\infty}{1 + \max\left(\|b\|_\infty, \|Ax\|_\infty\right)} \leq \varepsilon, \quad \eta_{\text{d}} = \frac{\|-Qx + A^*y + z - c\|_\infty}{1 + \max\left(\|c\|_\infty, \|A^*y\|_\infty, \|Qx\|_\infty\right)} \leq \varepsilon.$$

All other solvers are run with their respective default stopping conditions. We evaluate the performance of each solver using three accuracy levels: $\varepsilon = 10^{-4}$, $\varepsilon = 10^{-6}$, and $\varepsilon = 10^{-8}$. The numerical results for $\varepsilon = 10^{-4}$ are reported in Appendix B. Additionally, a time limit of 3600 seconds is imposed on all algorithms for each problem instance across all data sets, except for the extremely large-scale QAP relaxations.

Shifted Geometric Mean. To evaluate solver performance over multiple problems, we use the shifted geometric mean (SGM), as in Mittelmann's benchmarks. For a shift $\Delta = 10$, the SGM10 is defined as $\left(\prod_{i=1}^n (t_i + \Delta)\right)^{1/n} - \Delta$, where t_i denotes the solve time (or iteration count) in seconds for the i-th instance. Unsolved instances are assigned a time limit. Following PDQP [32], we exclude the time spent on data loading and preconditioning from the measured solve time. In HPR-QP, however, the time consumed by the power method is counted as part of the solve time, while PDQP does not include it. For SCS [37, 36] and CuClarabel [10], we record only the solve time and exclude any setup overhead. For Gurobi [17], we report the solve time as provided by the solver itself.

Absolute Performance Profile. We also evaluate solver performance using the absolute performance profile $f_s^a : \mathbb{R}_+ \to [0, 1]$, which represents the fraction of problems solved by solver s within time τ. It is defined as

$$f_s^a(\tau) = \frac{1}{N} \sum_p \mathcal{I}_{\leq \tau}(t_{p,s}),$$

where $\mathcal{I}_{\leq \tau}(u) = 1$ if $u \leq \tau$, and 0 otherwise.

4.2 Numerical Results on the Maros-Mészáros Data Set

We begin by evaluating all tested algorithms on the Maros-Mészáros data set [35], a standard benchmark for CQP problems comprising 137 instances.[4] Table 1 demonstrates the superior performance of HPR-QP over the HPR method applied to the primal reformulations (1.3) and (1.4), even when both use similar adaptive restart and penalty update strategies. Notably, our primal HPR variants also outperform PDQP in terms of SGM10 for both runtime and iteration count, further underscoring the efficiency and robustness of our dual HPR framework.

Table 1: Performance of HPR methods and PDQP on 15 Maros-Mészáros instances with tolerance 10^{-8}. HPR (p1) and HPR (p2) correspond to CCQP formulations (1.3) and (1.4), respectively.

Instance	HPR-QP		HPR (p1)		HPR (p2)		PDQP	
	Iter	Time	Iter	Time	Iter	Time	Iter	Time
CVXQP3_L	114 400	5.4	110 200	3.4	244 200	9.2	136 512	37.4
DUALC1	1 900	0.1	6 800	0.7	7 200	0.6	10 560	3.5
Q25FV47	215 100	32.4	379 400	37.8	531 600	49.9	891 552	245.6
QBANDM	23 500	1.4	136 600	5.9	213 900	9.9	139 968	37.6
QBRANDY	51 100	2.3	63 100	2.1	108 400	4.3	90 432	24.6
QCAPRI	687 700	30.0	3 577 300	121.9	4 539 300	161.2	1 939 104	524.5
QE226	27 000	1.3	67 900	2.8	115 400	4.2	157 824	43.2
QISRAEL	24 300	1.3	50 000	2.3	65 200	2.9	108 096	30.3
QSC205	14 700	1.3	14 800	1.2	21 100	1.5	58 656	16.1
QSCAGR25	20 100	1.1	59 200	2.7	61 800	2.4	62 400	17.5
QSCFXM3	277 300	12.7	2 123 400	73.4	1 284 100	47.0	428 544	119.7
QSEBA	111 900	6.0	160 500	8.4	165 200	8.1	623 520	166.8
QSHARE1B	72 200	3.5	2 020 400	67.3	2 324 300	83.1	312 672	81.6
QSHIP12L	70 600	4.5	63 200	2.6	75 500	3.5	135 936	38.4
QSIERRA	8 800	0.5	40 400	1.3	39 800	1.5	239 328	64.4
SGM10	46 562	5.0	129 068	11.0	162 336	12.8	177 259	55.3

We compare HPR-QP with PDQP, SCS, CuClarabel, and Gurobi on the Maros–Mészáros dataset—a relatively small-scale benchmark—with results summarized in Table 2 and visualized in Figure 1. Among all first-order methods tested, HPR-QP delivers the best overall performance: it solves more instances than PDQP at both 10^{-6} and 10^{-8} tolerances and significantly outperforms SCS in both robustness and efficiency. Specifically, HPR-QP achieves substantially lower SGM10 in runtime—approximately 3.1× faster at 10^{-6} and 3.4× faster at 10^{-8}—compared to PDQP. While CuClarabel achieves a lower SGM10 in runtime, it solves fewer problems than HPR-QP at the 10^{-8} tolerance.

4.3 Numerical Results on Synthetic CQP Problems

To complement our evaluation on the relatively small Maros-Mészáros dataset, we also generated 30 large-scale synthetic CQP instances across six problem classes following Section 8 of [42]. Table 8 in Appendix B details their dimensions and sparsity. The numerical performance of all tested solvers is summarized in Table 3 and visualized in Figure 2. At the tighter 10^{-8} accuracy level, HPR-QP successfully solves all synthetic instances within one hour. Its SGM10 in runtime is approximately 2.2× faster than the best-performing second-order solver, CuClarabel. When

[4] The instance `VALUES` is excluded from our evaluation, as Gurobi reports it to be non-convex.

Table 2: Numerical performance on 137 instances of the Maros-Mészáros data set (Tol. 10^{-6} and 10^{-8}).

Solver	10^{-6}		10^{-8}	
	SGM10 (Time)	Solved	SGM10 (Time)	Solved
HPR-QP	10.5	129	12.6	128
PDQP	33.1	125	42.5	124
SCS	126.0	103	165.0	93
CuClarabel	3.7	130	7.8	124
Gurobi	0.4	137	1.2	135

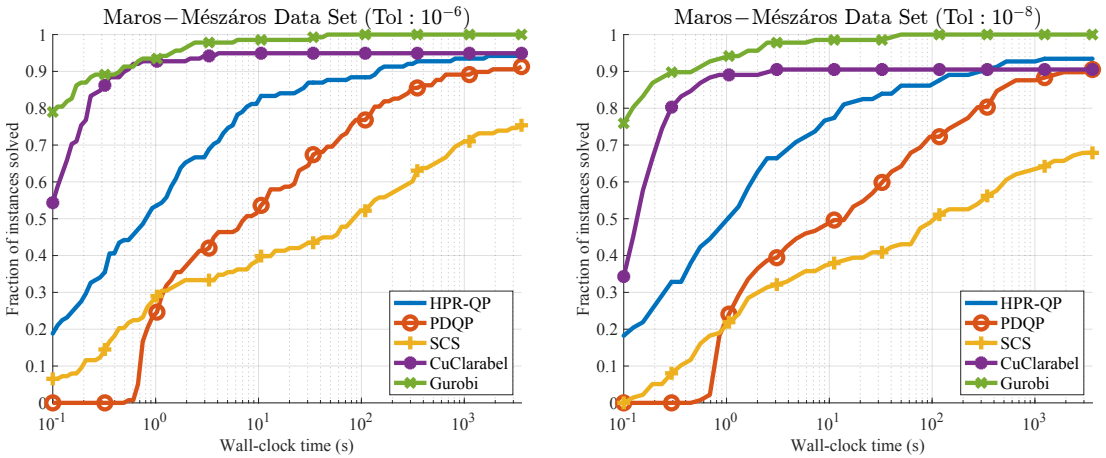


Figure 1: Absolute performance profiles of solvers on the 137 Maros-Mészáros QP instances

compared with first-order methods, HPR-QP still leads: as shown in Figure 2, it solves about 90% of the instances in approximately 100 seconds, whereas the next-best first-order solver, PDQP, requires nearly 1,000 seconds to achieve the same success rate. These results clearly demonstrate the superior efficiency and scalability of HPR-QP.

Table 3: Numerical performance on 30 randomly generated CQP instances (Tol. 10^{-6} and 10^{-8}).

Solver	10^{-6}		10^{-8}	
	SGM10 (Time)	Solved	SGM10 (Time)	Solved
HPR-QP	14.3	30	19.6	30
PDQP	51.9	28	63.8	27
SCS	781.5	13	847.7	12
CuClarabel	41.4	25	43.1	25
Gurobi	238.2	19	242.8	19

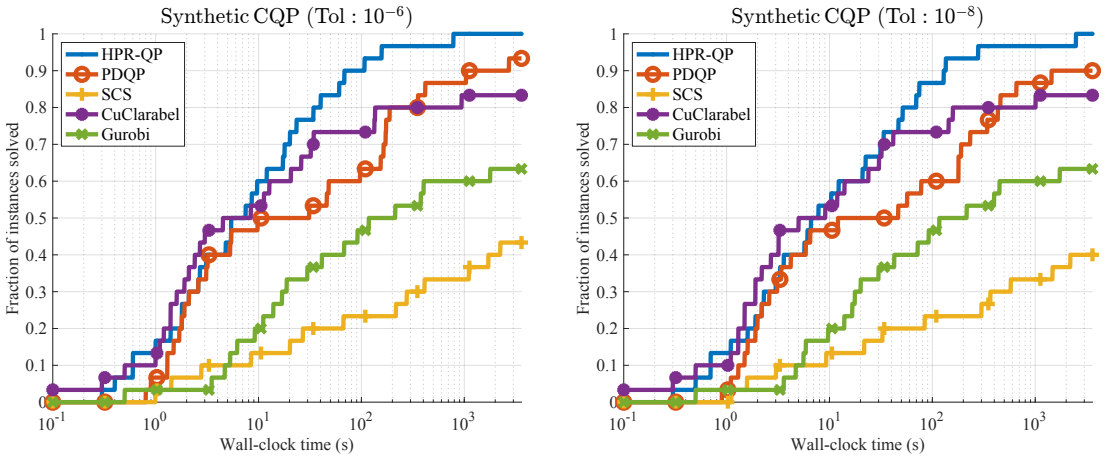


Figure 2: Absolute performance profiles of tested solvers on 30 synthetic CQP instances.

4.4 Numerical Results on Lasso Instances

Consider the following Lasso problem:

$$\min_{x\in\mathbb{R}^q}\left\{\frac{1}{2}\|\widehat{A}x-\widehat{b}\|^2+\lambda\|x\|_1\right\}, \tag{4.1}$$

where $\widehat{A}\in\mathbb{R}^{p\times q}$, $\widehat{b}\in\mathbb{R}^p$, and $\lambda>0$ is the regularization parameter. This problem can be equivalently reformulated as the following CQP:

$$\min_{(x,s,t)\in\mathbb{R}^q\times\mathbb{R}^p\times\mathbb{R}^q}\left\{\frac{1}{2}\|s\|^2+\lambda\sum_{i=1}^{q}t_i\;\middle|\;s=\widehat{A}x-\widehat{b},\ -t_i\le x_i\le t_i,\ i=1,\dots,q\right\}. \tag{4.2}$$

We evaluate performance on 11 Lasso instances from the UCI Machine Learning Repository (as used in [24]), originally sourced from the LIBSVM datasets [7]. Table 9 in Appendix B reports detailed problem dimensions and sparsity levels of the data matrix $\widehat{A}$. Numerical results are shown in Table 4. In terms of SGM10 in runtime at the 10^{-8} tolerance, HPR-QP significantly

outperforms both the first-order solver PDQP and the second-order solver CuClarabel. Specifically, HPR-QP is approximately 1.98× faster than CuClarabel and over 12× faster than PDQP, clearly demonstrating its superiority in both efficiency and scalability across solver categories.

Table 4: Numerical performance on 11 Lasso instances (Tol. 10^{-6} and 10^{-8}, $\lambda = 10^{-3}\|\widehat{A}^*\widehat{b}\|_\infty$). HPR-QP solves the original Lasso formulation (4.1), while other solvers are applied to its CQP reformulation (4.2). Abbreviations: 'T' = time-limit, 'F' = failure (e.g. unbounded or infeasible).

	HPR-QP		PDQP		SCS		CuClarabel		Gurobi	
Instance	10^{-6}	10^{-8}	10^{-6}	10^{-8}	10^{-6}	10^{-8}	10^{-6}	10^{-8}	10^{-6}	10^{-8}
abalone7	4.2	10.5	248.6	372.5	T	T	23.2	24.4	109.1	127.3
bodyfat7	1.0	1.2	27.8	33.3	T	T	2.0	2.2	29.8	30.8
E2006.test	0.1	0.2	1.2	1.3	T	T	10.5	15.4	8.7	9.0
E2006.train	0.4	0.7	1.8	1.9	F	F	114.0	116.0	272.8	277.8
housing7	11.0	22.6	83.5	123.3	T	T	5.5	5.7	123.5	125.9
log1p.E2006.test	5.0	7.0	1094.6	1416.9	T	T	183.0	196.0	107.1	137.0
log1p.E2006.train	13.9	17.3	2475.7	2983.2	T	T	335.0	361.0	593.7	878.8
mpg7	0.4	0.6	11.6	18.1	1200.0	2000.0	0.2	0.3	1.2	1.2
pyrim5	35.4	49.1	298.1	410.6	T	T	3.4	3.5	35.7	35.9
space_ga9	0.5	0.6	34.9	62.7	988.0	1210.0	6.1	6.7	33.4	38.1
triazines4	130.7	401.3	2546.0	3533.3	T	T	25.5	26.0	455.2	843.1
SGM10 (Time)	8.5	13.2	124.2	161.8	2898.0	3091.0	24.5	26.1	78.0	91.2

To further evaluate scalability, we generated 12 larger Lasso instances following Appendix A.5 of [42]. Numerical results are summarized in Table 5. For the relatively small instances ($p = 10^4$), HPR-QP solves the problems in under 0.5 seconds—faster than the second-best solver, CuClarabel, which takes about 2.5–3.8 seconds. On larger instances ($p \geq 2 \times 10^5$), both CuClarabel and Gurobi fail to solve the problems due to time limits or memory exhaustion. In contrast, HPR-QP remains robust and efficient. Against the first-order solver PDQP, HPR-QP consistently achieves speedups of approximately 3.9× to 6.3× in runtime at the 10^{-8} tolerance for these large-scale cases.

Table 5: Numerical performance on randomly generated Lasso instances (Tol. 10^{-6} and 10^{-8}, $\lambda = \frac{1}{5}\|A^*b\|_\infty$). HPR-QP solves the original Lasso formulation (4.1), while other solvers are applied to its CQP reformulation (4.2). 'T' = time-limit, 'M' = out-of-memory.

Size		HPR-QP		PDQP		SCS		CuClarabel		Gurobi	
p	q	10^{-6}	10^{-8}	10^{-6}	10^{-8}	10^{-6}	10^{-8}	10^{-6}	10^{-8}	10^{-6}	10^{-8}
10^4	5×10^5	0.5	0.5	6.6	8.7	T	T	2.3	2.5	22.4	23.3
10^4	10^6	0.2	0.3	12.2	17.6	T	T	3.5	3.8	38.5	40.3
2×10^5	5×10^6	5.8	9.0	38.3	51.4	T	T	M	M	T	T
2×10^5	10^7	16.5	24.7	119.1	155.5	T	T	M	M	T	T
4×10^5	5×10^6	10.7	14.8	60.1	76.9	T	T	M	M	M	M
4×10^5	10^7	32.0	47.8	182.8	238.8	T	T	M	M	M	M
6×10^5	5×10^6	14.2	21.9	76.6	86.7	T	T	M	M	M	M
6×10^5	10^7	47.6	71.2	257.5	328.7	T	T	M	M	M	M
8×10^5	5×10^6	16.7	22.8	93.0	120.4	T	T	M	M	M	M
8×10^5	10^7	63.1	89.2	336.2	420.5	T	T	M	M	M	M
10^6	5×10^6	20.7	25.8	122.5	152.3	T	T	M	M	M	M
10^6	10^7	78.7	117.7	M	M	T	T	M	M	M	M
SGM10 (Time)		18.5	25.5	113.2	139.3	3600.0	3600.0	1401.3	1405.3	1691.8	1701.0

4.5 Numerical Results on QAP Relaxation Problems

Given matrices $\widehat{A}, \widehat{B} \in \mathbb{S}^d$ (the space of $d \times d$ symmetric matrices), the QAP is defined as

$$\min_{X\in\mathbb{R}^{d\times d}} \left\{ \langle \mathrm{vec}(X), (\widehat{B}\otimes\widehat{A})\,\mathrm{vec}(X)\rangle \;\middle|\; X\in\{0,1\}^{d\times d},\ Xe=e,\ X^*e=e \right\},$$

where $\otimes$ denotes the Kronecker product, $\mathrm{vec}(X)$ is the vectorization of the matrix X, and $e \in \mathbb{R}^d$ is the vector of all ones. As shown in [1], a good lower bound for the above QAP can be obtained by solving the following CQP:

$$\min_{\mathrm{vec}(X)\in\mathbb{R}^{d^2}} \left\{ \langle \mathrm{vec}(X), \widehat{Q}\,\mathrm{vec}(X)\rangle \;\middle|\; (e^*\otimes I_d)\,\mathrm{vec}(X)=e,\ (I_d\otimes e^*)\,\mathrm{vec}(X)=e,\ \mathrm{vec}(X)\geq 0 \right\},$$

where the matrix $\widehat{Q} \in \mathbb{S}^{d^2}$ corresponds to the self-adjoint positive semidefinite linear operator $\widehat{\mathcal{Q}}$ defined by $\widehat{\mathcal{Q}}(X) := (\widehat{A}X\widehat{B} - SX - XT)\,\forall X \in \mathbb{R}^{d\times d}$. Equivalently, the matrix form is given by $\widehat{Q} = (\widehat{B}\otimes\widehat{A} - I\otimes S - T\otimes I)$, with $S, T \in \mathbb{S}^d$ computed as follows. Let $\widehat{A} = V_A D_A V_A^*$ and $\widehat{B} = V_B D_B V_B^*$ be the eigenvalue decompositions of $\widehat{A}$ and $\widehat{B}$, where $D_A = \mathrm{diag}(\alpha_1,\ldots,\alpha_d)$, $D_B = \mathrm{diag}(\beta_1,\ldots,\beta_d)$, with $\alpha_1 \geq \cdots \geq \alpha_d$ and $\beta_1 \leq \cdots \leq \beta_d$, and where $\mathrm{diag}(a_1,\ldots,a_d)$ denotes a diagonal matrix with entries $a_1,\ldots,a_d$ on the main diagonal. Let $(\bar{s},\bar{t})$ be the solution to the linear program: $\max\{\langle e, s+t\rangle \mid s_i + t_j \leq \alpha_i\beta_j,\ \forall i,j = 1,\ldots,d\}$, which admits a closed-form solution as described in [1]. Then, $S = V_A\,\mathrm{diag}(\bar{s})V_A^*, T = V_B\,\mathrm{diag}(\bar{t})V_B^*$.

We tested 36 instances with $50 \leq d \leq 256$ from the QAPLIB benchmark set [6].[5] The numerical results are summarized in Figure 3 and Table 6. Among the first-order solvers, Table 6 shows that HPR-QP significantly outperforms SCS, the second-best solver in this category, achieving a speedup of approximately 18.3× at 10^{-8} tolerance and 6.3× at 10^{-6} tolerance. Figure 3 further demonstrates that for 10^{-8} accuracy, HPR-QP solves around 90% of the instances within 20 seconds, while SCS requires nearly 1000 seconds to reach the same success rate.

[5] The instance sko90 is excluded, as Gurobi reports it to be non-convex.

Table 6: Numerical performance on 36 instances of QAP relaxations (Tol. 10^{-6} and 10^{-8}).

Solver	10^{-6}		10^{-8}	
	SGM10 (Time)	Solved	SGM10 (Time)	Solved
HPR-QP	1.8	36	4.7	36
PDQP	124.1	23	149.4	23
SCS	11.3	36	86.0	36
CuClarabel	13.6	33	114.9	22
Gurobi	24.8	36	26.8	36

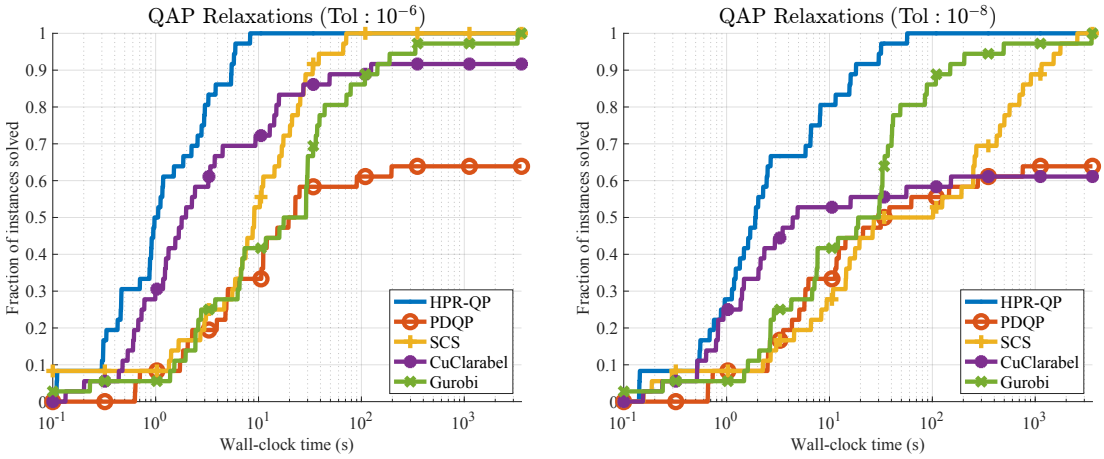


Figure 3: Absolute performance profiles of tested solvers on 36 instances of QAP relaxations.

To further evaluate the scalability of the solvers, we generate extremely large-scale synthetic QAP instances. Specifically, we uniformly sample d points in the unit square $[0,1)^2$, and define the matrix $\widehat{A}$ as their pairwise Euclidean distance matrix, i.e., $\widehat{A}_{ij} = \|(x_i, y_i) - (x_j, y_j)\|$. The matrix $\widehat{B}$ is constructed as a symmetric random matrix with entries drawn uniformly from $[0,1)$ and zero diagonal. Due to the prohibitive size of d and the limited memory, forming an explicit matrix representation of Q becomes infeasible. Consequently, we only report the results for HPR-QP in Table 7, highlighting its crucial advantage of operating without an explicit form of Q. Notably, HPR-QP successfully solves QAP relaxation instances up to $d = 8192$, demonstrating its exceptional scalability and efficiency in matrix-free settings.

Table 7: The runtime performance of HPR-QP on extremely large-scale QAP relaxations.

d	10^{-4}	10^{-6}	10^{-8}
256	0.8	3.0	6.0
512	1.3	4.0	9.3
1024	3.6	26.3	73.3
2048	73.0	291.0	700.0
4096	32.0	3640.0	9580.7
8192	188.6	61905.6	126547.9

5 Conclusion

In this paper, we proposed HPR-QP, a dual HPR method designed for solving large-scale CCQP problems. HPR-QP incorporates adaptive restart and penalty parameter update strategies to enhance convergence and robustness. Extensive numerical experiments on benchmark CCQP data sets demonstrate that HPR-QP delivers competitive performance across a wide range of accuracy requirements, particularly excelling in large-scale scenarios where second-order methods face limitations in scalability. Nonetheless, for small-scale problems, second-order methods may still offer superior efficiency due to their fast local convergence. As a potential direction for future work, integrating direct solvers into the HPR-QP framework to more efficiently handle the linear systems in subproblems could further improve its performance on smaller instances. Additionally, it would be interesting to explore hybrid strategies that combine HPR-QP with second-order methods on GPU, enabling more adaptive solutions across different problem scales.

References

[1] K. M. Anstreicher and N. W. Brixius. A new bound for the quadratic assignment problem based on convex quadratic programming. *Math. Program.*, 89:341–357, 2001.

[2] D. Applegate, M. Díaz, O. Hinder, H. Lu, M. Lubin, B. O'Donoghue, and W. Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. In *Advances in Neural Information Processing System*, volume 34, pages 20243–20257, 2021.

[3] D. Applegate, O. Hinder, H. Lu, and M. Lubin. Faster first-order primal-dual methods for linear programming using restarts and sharpness. *Math. Program.*, 201(1):133–184, 2023.

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Rev.*, 59(1):65–98, 2017.

[5] K. Bredies, E. Chenchene, D. A. Lorenz, and E. Naldi. Degenerate preconditioned proximal point algorithms. *SIAM J. Optim.*, 32(3):2376–2401, 2022.

[6] R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB–a quadratic assignment problem library. *J. Glob. Optim.*, 10:391–403, 1997.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):1–27, 2011.

[8] K. Chen, D. F. Sun, Y. Yuan, G. Zhang, and X. Zhao. HPR-LP: An implementation of an HPR method for solving linear programming. *arXiv preprint arXiv:2408.12179*, 2024.

[9] Y. Chen, G. Lan, and Y. Ouyang. Optimal primal-dual methods for a class of saddle point problems. *SIAM J. Optim.*, 24(4):1779–1814, 2014.

[10] Y. Chen, D. Tse, P. Nobel, P. Goulart, and S. Boyd. CuClarabel: GPU acceleration for a conic optimization solver. *arXiv preprint arXiv:2412.19027*, 2024.

[11] Q. Deng, Q. Feng, W. Gao, D. Ge, B. Jiang, Y. Jiang, J. Liu, T. Liu, C. Xue, Y. Ye, et al. An enhanced alternating direction method of multipliers-based interior point method for linear and conic optimization. *INFORMS J. Comput*, 2024.

[12] W. S. Dorn. Duality in quadratic programming. *Quart. Appl. Math.*, 18:155–162, 1960/61.

[13] J. Eckstein and D. P. Bertsekas. On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Program.*, 55(1):293–318, 1992.

[14] C. F. Gerald. *Applied Numerical Analysis*. Pearson Education India, 2004.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU press, 2013.

[16] P. J. Goulart and Y. Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv preprint arXiv:2405.12762*, 2024.

[17] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[18] B. Halpern. Fixed points of nonexpanding maps. *Bull. Am. Math. Soc.*, 73(6):957–961, 1967.

[19] D. Han, D. F. Sun, and L. Zhang. Linear rate convergence of the alternating direction method of multipliers for convex composite programming. *Math. Oper. Res.*, 43(2):622–637, 2018.

[20] Y. Huang, W. Zhang, H. Li, D. Ge, H. Liu, and Y. Ye. Restarted primal-dual hybrid conjugate gradient method for large-scale quadratic programming. *arXiv preprint arXiv:2405.16160*, 2024.

[21] IBM. IBM ILOG CPLEX Optimization Studio CPLEX User's Manual, 1987.

[22] M. Li, D. F. Sun, and K.-C. Toh. A convergent 3-block semi-proximal ADMM for convex minimization problems with one strongly convex block. *Asia-Pac. J. Oper. Res.*, 32(04):1550024, 2015.

[23] X. Li, D. F. Sun, and K.-C. Toh. A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions. *Math. Program.*, 155(1-2):333–373, 2016.

[24] X. Li, D. F. Sun, and K.-C. Toh. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM J. Optim.*, 28(1):433–458, 2018.

[25] X. Li, D. F. Sun, and K.-C. Toh. QSDPNAL: A two-phase augmented Lagrangian method for convex quadratic semidefinite programming. *Math. Program. Comput.*, 10:703–743, 2018.

[26] X. Li, D. F. Sun, and K.-C. Toh. A block symmetric Gauss–Seidel decomposition theorem for convex composite quadratic programming and its applications. *Math. Program.*, 175:395–418, 2019.

[27] L. Liang, X. Li, D. F. Sun, and K.-C. Toh. QPPAL: A two-phase proximal augmented Lagrangian method for high-dimensional convex quadratic programming problems. *ACM Trans. Math. Softw.*, 48(3):1–27, 2022.

[28] F. Lieder. On the convergence rate of the Halpern-iteration. *Optim. Lett.*, 15(2):405–418, 2021.

[29] T. Lin, S. Ma, Y. Ye, and S. Zhang. An ADMM-based interior-point method for large-scale linear programming. *Optim. Methods Softw.*, 36(2-3):389–424, 2021.

[30] Z. Lin, Z. Xiong, D. Ge, and Y. Ye. PDCS: A primal-dual large-scale conic programming solver with GPU enhancements. *arXiv preprint arXiv:2505.00311*, 2025.

[31] P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.*, 16(6):964–979, 1979.

[32] H. Lu and J. Yang. cuPDLP.jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia. *arXiv preprint arXiv:2311.12180*, 2023.

[33] H. Lu and J. Yang. A practical and optimal first-order method for large-scale convex quadratic programming. *arXiv preprint arXiv:2311.07710*, 2023.

[34] H. Lu, J. Yang, H. Hu, Q. Huangfu, J. Liu, T. Liu, Y. Ye, C. Zhang, and D. Ge. cuPDLP-C: A strengthened implementation of cuPDLP for linear programming by C language. *arXiv preprint arXiv:2312.14832*, 2023.

[35] I. Maros and C. Mészáros. A repository of convex quadratic programming problems. *Optim. Methods Softw.*, 11(1-4):671–681, 1999.

[36] B. O'Donoghue. Operator splitting for a homogeneous embedding of the linear complementarity problem. *SIAM J. Optim.*, 31(3):1999–2023, 2021.

[37] B. O'donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *J. Optim. Theory Appl.*, 169:1042–1068, 2016.

[38] T. Pock and A. Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*, pages 1762–1769. IEEE, 2011.

[39] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[40] D. Ruiz. A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical report, Rutherford Appleton Laboratory, 2001.

[41] S. Sabach and S. Shtern. A first order method for solving convex bilevel optimization problems. *SIAM J. Optim.*, 27(2):640–660, 2017.

[42] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *Math. Program. Comput.*, 12(4):637–672, 2020.

[43] D. F. Sun, Y. Yuan, G. Zhang, and X. Zhao. Accelerating preconditioned ADMM via degenerate proximal point mappings. *SIAM J. Optim.*, 35(2):1165–1193, 2025.

[44] P. Wolfe. A duality theorem for non-linear programming. *Quart. Appl. Math.*, 19:239–244, 1961.

[45] B. Yang, X. Zhao, X. Li, and D. F. Sun. An accelerated proximal alternating direction method of multipliers for optimal decentralized control of uncertain systems. *J. Optim. Theory Appl.*, 204(1):9, 2025.

[46] G. Zhang, Z. Gu, Y. Yuan, and D. F. Sun. HOT: An efficient Halpern accelerating algorithm for optimal transport problems. *IEEE Trans. Pattern Anal. Mach. Intell. XXX, in print.*, 2025.

[47] G. Zhang, Y. Yuan, and D. F. Sun. An efficient HPR algorithm for the Wasserstein barycenter problem with $O(\mathrm{Dim}(\mathrm{P})/\varepsilon)$ computational complexity. *arXiv preprint arXiv:2211.14881*, 2022.

A An HPR Method for the Primal Form of CCQP

A.1 An HPR Method Applied to the Primal Reformulation (1.3)

Recall the following primal reformulation of CCQP problems:

$$\min_{(x,s)\in\mathbb{R}^n\times\mathbb{R}^m}\left\{\frac{1}{2}\langle x,Qx\rangle+\langle c,x\rangle+\phi(x)+\delta_{\mathcal{K}}(s)\mid Ax=s\right\}. \tag{A.1}$$

Given $\sigma>0$, we define the augmented Lagrangian function $L_\sigma(x,s;y)$ associated with problem (A.1) for any $(x,s,y)\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^m$ as follows:

$$L_\sigma(x,s;y)=\frac{1}{2}\langle x,Qx\rangle+\langle c,x\rangle+\phi(x)+\delta_{\mathcal{K}}(s)+\langle s-Ax,y\rangle+\frac{\sigma}{2}\|Ax-s\|^2.$$

For the sake of notational simplicity, we represent the tuple (x,s,y) as u, and we denote the set $\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^m$ by $\mathcal{U}$. Then, an HPR method [43] for solving problem (A.1) is presented in Algorithm 6:

Algorithm 6 An HPR method for solving the primal reformulation (A.1)

1: Input: Choose $u^0=(x^0,s^0,y^0)\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^m$. Set parameters $\sigma>0$. Let $\mathcal{S}_x=\lambda_Q I_n-Q+\sigma(\lambda_A I_n-A^*A)$. Denote $u=(x,s,y)$ and $\bar{u}=(\bar{x},\bar{s},\bar{y})$.
2: **for** $k=0,1,2,\ldots$ **do**
3: Step 1. $\bar{s}^{k+1}=\arg\min_{s\in\mathbb{R}^m}\{L_\sigma(x^k,s;y^k)\}$;
4: Step 2. $\bar{y}^{k+1}=y^k+\sigma(\bar{s}^{k+1}-Ax^k)$;
5: Step 3. $\bar{x}^{k+1}=\arg\min_{x\in\mathbb{R}^n}\left\{L_\sigma(x,\bar{s}^{k+1};\bar{y}^{k+1})+\frac{1}{2}\|x-x^k\|^2_{\mathcal{S}_x}\right\}$;
6: Step 4. $\hat{u}^{k+1}=2\bar{u}^{k+1}-u^k$;
7: Step 5. $u^{k+1}=\frac{1}{k+2}u^0+\frac{k+1}{k+2}\hat{u}^{k+1}$;
8: **end for**

A.2 An HPR Method Applied to the Primal Reformulation (1.4)

Recall the following primal reformulation of CCQP problems:

$$\min_{(x,s,v)\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n}\left\{\frac{1}{2}\langle v,Qv\rangle+\langle c,x\rangle+\phi(x)+\delta_{\mathcal{K}}(s)\mid Ax=s,x=v\right\}. \tag{A.2}$$

Given $\sigma>0$, we define the augmented Lagrangian function $L_\sigma(x,s,v;y)$ associated with problem (A.2) for any $(x,s,v,y,t)\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n$ as follows:

$$L_\sigma(x,s,v;y,t)=\frac{1}{2}\langle v,Qv\rangle+\langle c,x\rangle+\phi(x)+\delta_{\mathcal{K}}(s)+\langle s-Ax,y\rangle+\langle v-x,t\rangle+\frac{\sigma}{2}\|Ax-s\|^2+\frac{\sigma}{2}\|x-v\|^2.$$

For the sake of notational simplicity, we represent the tuple (x,s,v,y,t) as u, and we denote the set $\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n$ by $\mathcal{U}$. Then, an HPR method [43] for solving problem (A.2) can be presented in Algorithm 7:

Algorithm 7 An HPR method for solving the primal reformulation (A.2)

Input: Choose $u^0 = (x^0, s^0, v^0, y^0, t^0) \in \mathcal{U}$. Set parameters $\sigma > 0$. Let $\mathcal{S}_v = \lambda_Q I_n - Q$ and $\mathcal{S}_x = \sigma(\lambda_A I_n - A^*A)$. Denote $u = (x, s, v, y, t)$ and $\bar{u} = (\bar{x}, \bar{s}, \bar{v}, \bar{y}, \bar{t})$.

for $k = 0, 1, 2, \ldots$ **do**

Step 1. $(\bar{s}^{k+1}, \bar{v}^{k+1}) = \underset{(s,v)\in\mathbb{R}^m\times\mathbb{R}^n}{\arg\min} \left\{ L_\sigma(x^k, s, v; y^k, t^k) + \frac{1}{2}\|v - v^k\|^2_{\mathcal{S}_v} \right\}$;

Step 2.1. $\bar{y}^{k+1} = y^k + \sigma(\bar{s}^{k+1} - Ax^k)$;

Step 2.2. $\bar{t}^{k+1} = t^k + \sigma(\bar{v}^{k+1} - x^k)$;

Step 3. $\bar{x}^{k+1} = \underset{x\in\mathbb{R}^n}{\arg\min} \left\{ L_\sigma(x, \bar{s}^{k+1}, \bar{v}^{k+1}; \bar{y}^{k+1}, \bar{t}^{k+1}) + \frac{1}{2}\|x - x^k\|^2_{\mathcal{S}_x} \right\}$;

Step 4. $\hat{u}^{k+1} = 2\bar{u}^{k+1} - u^k$;

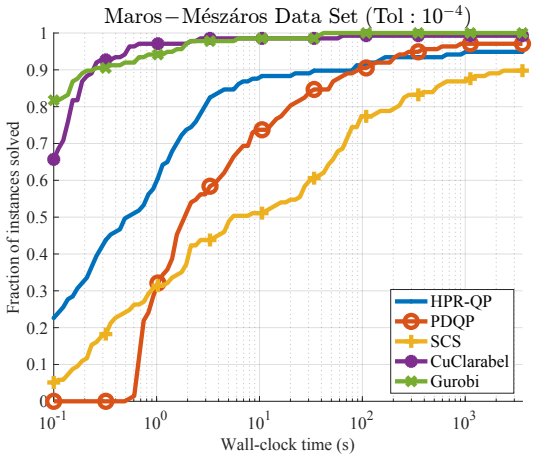
Step 5. $u^{k+1} = \frac{1}{k+2}u^0 + \frac{k+1}{k+2}\hat{u}^{k+1}$;

end for

B Additional Numerical Results

Solver	SGM10 (Time)	Solved
HPR-QP	7.4	130
PDQP	11.3	133
SCS	38.1	123
CuClarabel	0.7	136
Gurobi	0.4	137

(a) SGM10 and number solved



(b) Absolute performance profiles

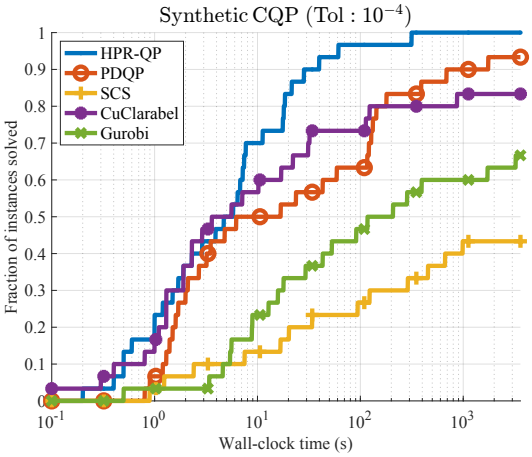
Figure 4: Numerical performance of tested solvers on 137 instances of the Maros-Mészáros data set (Tol. 10^{-4}).

Table 8: Problem dimensions and sparsity of matrices A and Q in the synthetic CQP instances.

Instance	Rows	Cols	nnz(A)	nnz(Q)
random_1	5.00×10^5	5.00×10^4	1.00×10^6	2.50×10^5
random_2	1.00×10^6	1.00×10^5	2.00×10^6	5.01×10^5
random_3	5.00×10^6	5.00×10^5	1.00×10^7	2.50×10^6
random_4	1.00×10^7	1.00×10^6	2.00×10^7	5.00×10^6
random_5	5.00×10^7	5.00×10^6	1.00×10^8	2.50×10^7
equality_1	5.00×10^3	5.00×10^3	1.99×10^5	6.80×10^6
equality_2	5.00×10^3	1.00×10^4	2.00×10^5	1.47×10^7
equality_3	1.00×10^4	2.00×10^4	4.00×10^5	3.07×10^7
equality_4	2.50×10^4	5.00×10^4	1.00×10^6	7.87×10^7
equality_5	5.00×10^4	1.00×10^5	2.00×10^6	1.59×10^8
control_1	2.20×10^3	3.20×10^3	6.02×10^5	4.24×10^4
control_2	5.50×10^3	8.00×10^3	3.76×10^6	2.56×10^5
control_3	1.10×10^4	1.60×10^4	1.50×10^7	1.01×10^6
control_4	1.65×10^4	2.40×10^4	3.38×10^7	2.27×10^6
control_5	2.20×10^4	3.20×10^4	6.00×10^7	4.02×10^6
portfolio_1	4.01×10^2	4.04×10^4	8.04×10^6	4.04×10^4
portfolio_2	5.01×10^2	5.05×10^4	1.26×10^7	5.05×10^4
portfolio_3	6.01×10^2	6.06×10^4	1.81×10^7	6.06×10^4
portfolio_4	7.01×10^2	7.07×10^4	2.46×10^7	7.07×10^4
portfolio_5	8.01×10^2	8.08×10^4	3.21×10^7	8.08×10^4
huber_1	5.00×10^5	1.51×10^6	1.60×10^6	5.00×10^5
huber_2	1.00×10^6	3.01×10^6	3.20×10^6	1.00×10^6
huber_3	2.00×10^6	6.02×10^6	6.40×10^6	2.00×10^6
huber_4	5.00×10^6	1.51×10^7	1.60×10^7	5.00×10^6
huber_5	1.00×10^7	3.01×10^7	3.20×10^7	1.00×10^7
svm_1	1.00×10^6	1.01×10^6	1.40×10^6	1.00×10^4
svm_2	2.00×10^6	2.02×10^6	2.80×10^6	2.00×10^4
svm_3	5.00×10^6	5.05×10^6	7.00×10^6	5.00×10^4
svm_4	1.00×10^7	1.01×10^7	1.40×10^7	1.00×10^5
svm_5	2.00×10^7	2.02×10^7	2.80×10^7	2.00×10^5

Solver	SGM10 (Time)	Solved
HPR-QP	8.6	30
PDQP	42.6	28
SCS	648.9	13
CuClarabel	39.1	25
Gurobi	230.4	20

(a) SGM10 and number of problems solved



(b) Absolute performance profiles

Figure 5: Numerical performance of tested solvers on 30 synthetic CQP problems (Tol. 10^{-4}).

Table 9: Dimensions and number of nonzeros in $\widehat{A}$ for UCI Lasso instances.

Instance	p	q	nnz($\widehat{A}$)
abalone7	4 177	6 435	22 873 422
bodyfat7	252	116 280	29 302 560
E2006.test	3 308	150 358	4 559 533
E2006.train	16 087	150 360	19 971 015
housing7	506	77 520	39 225 120
log1p.E2006.test	3 308	4 272 226	22 474 250
log1p.E2006.train	16 087	4 272 227	96 731 839
mpg7	392	3 432	1 174 932
pyrim5	74	201 376	8 054 057
space_ga9	3 107	5 005	15 550 535
triazines4	186	635 376	77 638 169

Table 10: Numerical performance on 11 Lasso instances (Tol. 10^{-4}). HPR-QP solves the original Lasso (4.1), others solve the CQP reformulation (4.2). 'T' = time-limit, 'F' = failure.

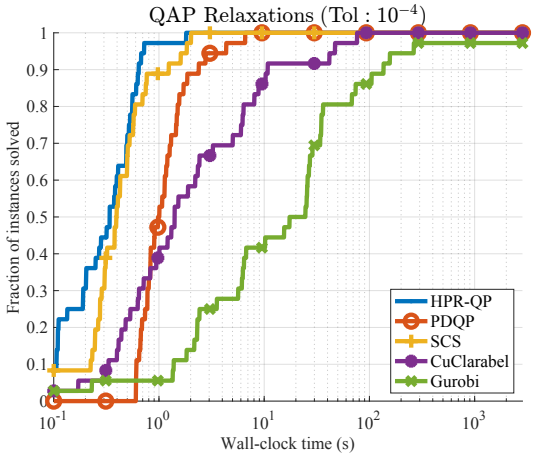
Instance	HPR-QP	PDQP	SCS	CuClarabel	Gurobi
abalone7	1.4	78.3	1850	21.8	85.8
bodyfat7	0.9	21.8	T	1.6	27.5
E2006.test	0.1	1.2	132.0	9.1	8.3
E2006.train	0.3	1.7	F	98.8	260.3
housing7	3.9	44.7	T	5.2	112.5
log1p.E2006.test	3.8	783.4	T	170.0	120.4
log1p.E2006.train	11.7	1830.4	T	310.0	551.2
mpg7	0.1	6.5	411.0	0.2	1.1
pyrim5	3.2	90.1	T	3.2	33.7
space_ga9	0.3	13.6	332.0	5.5	31.0
triazines4	57.0	1286.9	T	25.3	310.1
SGM10 (Time)	4.3	74.1	1776.0	22.9	71.3

Table 11: Numerical performance on randomly generated Lasso instances (Tol. 10^{-4}). HPR-QP solves the original Lasso (4.1), others solve the CQP reformulation (4.2). 'T' = time-limit, 'M' = out-of-memory.

p	q	HPR-QP	PDQP	SCS	CuClarabel	Gurobi
10^4	5×10^5	0.4	4.7	T	2.1	21.9
10^4	10^6	0.1	5.5	T	3.0	36.7
2×10^5	5×10^6	3.6	31.2	T	M	T
2×10^5	10^7	8.4	94.1	T	M	T
4×10^5	5×10^6	5.5	49.5	T	M	M
4×10^5	10^7	16.2	147.7	T	M	M
6×10^5	5×10^6	8.1	62.6	T	M	M
6×10^5	10^7	27.9	186.2	T	M	M
8×10^5	5×10^6	10.6	78.0	T	M	M
8×10^5	10^7	31.8	240.2	T	M	M
10^6	5×10^6	13.1	92.8	T	M	M
10^6	10^7	46.1	M	T	M	M
SGM10 (Time)		11.2	90.6	3600.0	1395.2	1684.4

Solver	SGM10 (Time)	Solved
HPR-QP	0.4	36
PDQP	1.3	36
SCS	0.5	36
CuClarabel	4.2	36
Gurobi	22.8	36

(a) SGM10 and number of problems solved



(b) Absolute performance profiles

Figure 6: Numerical performance of tested solvers on 36 QAP relaxations (Tol. 10^{-4}).