Noname manuscript No.
(will be inserted by the editor)

HPR-LP: An implementation of an HPR method for solving linear programming

Kaihuang Chen · Defeng Sun · Yancheng Yuan · Guojun Zhang · Xinyuan Zhao

the date of receipt and acceptance should be inserted later

Abstract In this paper, we introduce an HPR-LP solver, an implementation of a Halpern Peaceman-Rachford (HPR) method with semi-proximal terms for solving linear programming (LP). The HPR method enjoys the iteration complexity of $O(1/k)$ in terms of the Karush-Kuhn-Tucker residual and the objective error. Based on the complexity results, we design an adaptive strategy of restart and penalty parameter update to improve the efficiency and robustness of the HPR method. We conduct extensive numerical experiments on different LP benchmark datasets using NVIDIA A100-SXM4-80GB GPU in different stopping tolerances. Our solver's Julia version achieves a **2.39x** to **5.70x** speedup measured by SGM10 on benchmark datasets with presolve (**2.03x** to **4.06x** without presolve) over the award-winning solver PDLP with the tolerance of 10^{-8}.

Kaihuang Chen
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
E-mail: kaihuang.chen@connect.polyu.hk

Defeng Sun (✉)
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
E-mail: defeng.sun@polyu.edu.hk

Yancheng Yuan
Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
E-mail: yancheng.yuan@polyu.edu.hk

Guojun Zhang
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
E-mail: guojun.zhang@connect.polyu.hk

Xinyuan Zhao
Department of Mathematics, Beijing University of Technology, Beijing, P.R. China
E-mail: xyzhao@bjut.edu.cn

Keywords Linear programming · Halpern Peaceman-Rachford · Alternating direction method of multipliers · Acceleration · Complexity analysis

Mathematics Subject Classification (2020) 90C05 · 90C06 · 90C25 · 65Y20

1 Introduction

In this paper, we introduce how to implement a Halpern Peaceman-Rachford (HPR) method with semi-proximal terms [42] to solve the following linear programming (LP) problems:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \langle c, x\rangle \\ \text{s.t. } & A_1 x = b_1 \\ & A_2 x \geq b_2 \\ & x \in C, \end{aligned} \tag{1}$$

where $A_1 \in \mathbb{R}^{m_1\times n}$, $A_2 \in \mathbb{R}^{m_2\times n}$, $b_1 \in \mathbb{R}^{m_1}$, $b_2 \in \mathbb{R}^{m_2}$, $c \in \mathbb{R}^n$, and $C := \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$ with given vectors $l \in (\mathbb{R}\cup\{-\infty\})^n$ and $u \in (\mathbb{R}\cup\{+\infty\})^n$. Let $A = [A_1; A_2] \in \mathbb{R}^{m\times n}$ with $m = m_1 + m_2$, and $b = [b_1; b_2] \in \mathbb{R}^m$. In this paper, we assume that A is a non-zero matrix. Then, the dual of problem (1) is given by

$$\begin{aligned} \min_{y\in\mathbb{R}^m, z\in\mathbb{R}^n} \quad & -\langle b, y\rangle + \delta_D(y) + \delta_C^*(-z) \\ \text{s.t. } & A^* y + z = c, \end{aligned} \tag{2}$$

where $\delta_D(\cdot)$ is the indicator function over $D := \{y = (y_1, y_2) \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2}_+\}$ and $\delta_C^*(\cdot)$ is the conjugate of $\delta_C(\cdot)$.

Traditionally, the commercial solvers for LP [19,22] are based on the simplex methods and interior point methods. In recent years, first-order methods (FOMs) have gained increasing attention for solving large-scale LP problems due to their low iteration cost and ease of parallelization [2,3,4,12,26,29,33,34,41]. Notably, Applegate et al. [2,3] developed an award-winning solver PDLP[1], which employed the primal-dual hybrid gradient (PDHG) method [47] as its base algorithm, equipped with several effective implementation techniques, including the use of the ergodic iterate as a restart point[2], an update rule for the penalty parameter σ, and a line search strategy. The GPU implementation of PDLP (cuPDLP.jl [28] and cuPDLP-c [30]) has shown some advantages over commercial LP solvers like Gurobi [19] and COPT [16] for solving large-scale LP problems. It is worth noting that the ergodic sequence of the semi-proximal

[1] The authors of [2,3] were awarded the Beale–Orchard-Hays Prize for Excellence in Computational Mathematical Programming at the 25th International Symposium on Mathematical Programming (https://ismp2024.gerad.ca/), July 21-26, 2024, Montréal, Canada.

[2] In the GPU implementation (e.g., [28] and [30]), the restart point is selected based on the weighted KKT residual, utilizing either the last iterate or the ergodic iterate. For theoretical analysis, the ergodic iterate is used as the restart point [3].

ADMM (sPADMM) [15], including PDHG [9,14], achieves an $O(1/k)$ iteration complexity with respect to the objective error and the feasibility violation [11], where k is the iteration number. To the best of our knowledge, it is still unknown whether the ergodic sequence of sPADMM can achieve an $O(1/k)$ iteration complexity in terms of the Karush-Kuhn-Tucker (KKT) residual for LP.[3]

Recently, some progress in complexity results has been achieved in accelerating the preconditioned (semi-proximal) PR [42,44,46] using the Halpern iteration [20,25,39]. In particular, Zhang et al. [46] applied the Halpern iteration to the PR splitting method [13,27], developing the HPR method without proximal terms.[4] This HPR method achieves an iteration complexity of $O(1/k)$ for the KKT residual and the objective error. Subsequently, Yang et al. [44] reformulated the proximal PR method as a proximal point method (PPM) with a positive definite preconditioner, developing an HPR method with proximal terms via the Halpern iteration that obtains the $O(1/k)$ iteration complexity for the weighted fixed-point residual. Finally, Sun et al. [42] reformulated the semi-proximal PR method into a degenerate PPM (dPPM) with a positive semidefinite preconditioner [7] and applied the Halpern iteration to this dPPM, resulting in an HPR method with semi-proximal terms that enjoys the desired $O(1/k)$ iteration complexity for the KKT residual and the objective error.[5] Compared to existing algorithms for solving LP, the HPR method [42, 46] offers a key theoretical advantage: an $O(1/k)$ iteration complexity in terms of the KKT residual, which motivates us to explore the potential of HPR for solving large-scale LP problems.

The main purpose of this paper is to introduce a solver called HPR-LP for solving LP problems. The main features of our solver are highlighted below.

1. Based on the iteration complexity of $O(1/k)$ in terms of the KKT residual, in our HPR-LP solver we integrate a restart strategy and an update rule of penalty parameter σ into the HPR method with semi-proximal terms for solving large-scale LP problems.
2. We test the numerical performance of the HPR-LP solver on different LP benchmark datasets using NVIDIA A100-SXM4-80GB GPU in different stopping tolerances. The Julia version of our solver achieves a **2.39x** to **5.70x** speedup measured by SGM10 on benchmark datasets with presolve (**2.03x** to **4.06x** without presolve) over the award-winning solver PDLP

[3] Shen and Pan [40] extended Monteiro and Svaiter's ergodic $O(1/k)$ iteration complexity result with respect to the ε-subdifferential residual [31] from ADMM to sPADMM, which can yield an $O(1/\sqrt{k})$ iteration complexity for the KKT residual of LP problems.

[4] The HPR method without proximal terms is equivalent to Kim's accelerated proximal point method (PPM) applied to the Douglas-Rachford method in view of the relationship between Halpern's iteration and Kim's accelerated PPM [23,38].

[5] The Halpern PDHG for LP proposed in [29] corresponds to a special case of the HPR method in [42] with one proximal term to be strongly convex and $\rho = 1$. Inspired by our work, Lu and Yang [29], in their revised version, extended their approach by introducing the reflected restarted Halpern PDHG (r^2HPDHG), which adopts $\rho = 2$ within the framework of the HPR method in [42]. By incorporating similar algorithmic enhancements as cuPDLP [28], r^2HPDHG achieves a 1.27x to 1.33x speedup over cuPDLP in their numerical experiments.

with the tolerance of 10^{-8}. Moreover, for the same accuracy, HPR-LP successfully solves more problems than cuPDLP.jl does.

The remaining parts of this paper are organized as follows. In Section 2, we briefly introduce the base algorithm, an HPR method with semi-proximal terms, for solving LP. Then, in Section 3, we discuss the implementation of HPR-LP, which incorporates a restart strategy and an update rule for the penalty parameter. Section 4 provides extensive numerical results on different LP benchmark sets. Finally, we conclude the paper in Section 5.

Notation. Let $\mathbb{R}^n$ be the n-dimensional real Euclidean space equipped with an inner product $\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$. We denote the nonnegative orthant of $\mathbb{R}^n$ as $\mathbb{R}^n_+$. For a matrix $A \in \mathbb{R}^{m\times n}$, we denote its transpose by A^* and its spectral norm by $\|A\| := \sqrt{\lambda_1(AA^*)}$, where $\lambda_1(AA^*)$ represents the largest eigenvalue of the symmetric matrix AA^*. Additionally, 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 $x \in \mathbb{R}^n$. For any convex function $f : \mathbb{R}^n \to (-\infty, +\infty]$, we denote its effective domain as $\mathrm{dom}(f) := \{x \in \mathbb{R}^n \mid f(x) < +\infty\}$, its subdifferential as $\partial f(\cdot)$, its conjugate as $f^*(z) := \sup_{x\in\mathbb{R}^n}\{\langle x, z\rangle - f(x)\}$, $z \in \mathbb{R}^n$, and its proximal mapping as $\mathrm{Prox}_f(x) := \arg\min_{z\in\mathbb{R}^n}\left\{f(z) + \frac{1}{2}\|z - x\|^2\right\}$, $x \in \mathbb{R}^n$. Let $C \subseteq \mathbb{R}^n$ be a convex set. Denote the indicator function over C by $\delta_C(\cdot)$, i.e., for any $x \in \mathbb{R}^n$, $\delta_C(x) = 0$ if $x \in C$, and $\delta_C(x) = +\infty$ if $x \notin C$. We write the distance of $x \in \mathbb{R}^n$ to C as $\mathrm{dist}(x, C) := \inf_{z\in C}\|z - x\|$. For a given closed convex set $C \subseteq \mathbb{R}^n$ and a given point $x \in \mathbb{R}^n$, the Euclidean projection of x onto C is denoted by $\Pi_C(x) := \arg\min\{\|x - z\| \mid z \in C\}$. Moreover, for any $x \in C$, we use $\mathcal{N}_C(x)$ to denote the normal cone of C at x.

2 Preliminaries

In this section, we introduce an HPR method with semi-proximal terms for solving LP problems. According to [36, Corollary 28.3.1], we know that $(y^*, z^*) \in \mathbb{R}^m \times \mathbb{R}^n$ is an optimal solution to problem (2) if and only if there exists $x^* \in \mathbb{R}^n$ such that (y^*, z^*, x^*) satisfies the following KKT system:

$$0 \in Ax^* - b + \mathcal{N}_D(y^*), \quad 0 \in z^* + \mathcal{N}_C(x^*), \quad A^*y^* + z^* - c = 0. \tag{3}$$

For any $(y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$, the augmented Lagrangian function associated with the dual problem (2) is defined as

$$L_\sigma(y, z; x) := -\langle b, y\rangle + \delta_D(y) + \delta_C^*(-z) + \langle x, A^*y + z - c\rangle + \frac{\sigma}{2}\|A^*y + z - c\|^2,$$

where $\sigma > 0$ is a given penalty parameter. For notational convenience, let $w := (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$. Then, an HPR method with semi-proximal terms proposed in [42] for solving problems (1) and (2) is presented in Algorithm 1.

Algorithm 1 corresponds to the accelerated preconditioned (semi-proximal) ADMM (pADMM) introduced in [42] with $\alpha = 2$, where Step 5 is the Halpern

Algorithm 1 An HPR method with semi-proximal terms for the problem (2)

1: **Input:** Set the penalty parameter $\sigma > 0$. Let $\mathcal{T}_1 : \mathbb{R}^m \to \mathbb{R}^m$ be a self-adjoint positive semidefinite linear operator such that $\mathcal{T}_1 + \sigma AA^*$ is positive definite. Denote $w = (y, z, x)$ and $\bar{w} = (\bar{y}, \bar{z}, \bar{x})$. Choose an initial point $w^0 = (y^0, z^0, x^0) \in D \times \mathbb{R}^n \times \mathbb{R}^n$.

2: **for** $k = 0, 1, ...,$ **do**

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

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

5: Step 3. $\bar{y}^{k+1} = \arg\min_{y \in \mathbb{R}^m} \left\{ L_\sigma \left(y, \bar{z}^{k+1}; \bar{x}^{k+1} \right) + \frac{\sigma}{2} \|y - y^k\|^2_{\mathcal{T}_1} \right\}$;

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

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

8: **end for**

9: **Output:** Iteration sequence $\{\bar{w}^k\}$.

iteration with a stepsize of $\frac{1}{k+2}$ [20,25]. We refer to it as an HPR method with semi-proximal terms because, for the convex optimization problem (27), the Halpern accelerating pADMM (without proximal terms) [42] is equivalent to the HPR method without proximal terms [46], whose proof can be found in Appendix A. Now, to discuss the global convergence of the HPR method with semi-proximal terms presented in Algorithm 1, we make the following assumption:

Assumption 1 *There exists a vector* $(y^*, z^*, x^*) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ *satisfying the KKT system* (3).

Under Assumption 1, solving problems (1) and (2) is equivalent to finding a $w^* \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ such that $0 \in \mathcal{T} w^*$, where the maximal monotone operator $\mathcal{T}$ is defined by

$$\mathcal{T} w = \begin{pmatrix} -b + \mathcal{N}_D(y) + Ax \\ -\partial \delta^*_C(-z) + x \\ c - A^* y - z \end{pmatrix}, \quad \forall w = (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n. \tag{4}$$

Consider the following self-adjoint linear operator $\mathcal{M} : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$,

$$\mathcal{M} = \begin{bmatrix} \sigma AA^* + \sigma \mathcal{T}_1 & 0 & A \\ 0 & 0 & 0 \\ A^* & 0 & \frac{1}{\sigma} I_n \end{bmatrix}, \tag{5}$$

where I_n denotes the identity matrix in $\mathbb{R}^{n \times n}$. We can establish the equivalence between the HPR method with semi-proximal terms and the accelerated dPPM [42] in the following proposition.

Proposition 1 *Consider the operators* $\mathcal{T}$ *defined in* (4) *and* $\mathcal{M}$ *defined in* (5). *Then the sequence* $\{w^k\}$ *generated by the HPR method with semi-proximal*

terms in Algorithm 1 coincides with the sequence $\{w^k\}$ *generated by the following accelerated dPPM*

$$\begin{cases} \bar{w}^{k+1} = (\mathcal{M}+\mathcal{T})^{-1}\mathcal{M}w^k, \\ \hat{w}^{k+1} = 2\bar{w}^{k+1} - w^k, \\ w^{k+1} = \dfrac{1}{k+2}w^0 + \dfrac{k+1}{k+2}\hat{w}^{k+1}, \end{cases}$$

with the same initial point $w^0 \in D \times \mathbb{R}^n \times \mathbb{R}^n$.

Proof This result can be derived similarly from [42, Proposition 3.2]. □

By utilizing the equivalence in Proposition 1, we can establish the global convergence of the HPR method with semi-proximal terms, as presented in the following proposition.

Proposition 2 (Corollary 3.5 in [42]) *Suppose that Assumption 1 holds. Then the sequence* $\{\bar{w}^k\} = \{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ *generated by the HPR method with semi-proximal terms in Algorithm 1 converges to the point* $w^* = (y^*, z^*, x^*)$*, where* (y^*, z^*) *solves problem* (2) *and* x^* *solves problem* (1).

Moreover, the equivalence in Proposition 1, together with the iteration complexity of the Halpern iteration [25], yields the following iteration complexity for the HPR method with semi-proximal terms:

Proposition 3 (Proposition 2.9 in [42]) *Suppose that Assumption 1 holds. Then the sequences* $\{w^k\}$ *and* $\{\hat{w}^k\}$ *generated by the HPR method with semi-proximal terms in Algorithm 1 satisfy*

$$\|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} \le \frac{2\left\|w^0 - w^*\right\|_{\mathcal{M}}}{k+1}, \quad \forall k \ge 0 \text{ and } w^* \in \mathcal{T}^{-1}(0). \tag{6}$$

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 (3), as introduced in [21]:

$$\mathcal{R}(w) = \begin{pmatrix} y - \Pi_D(y - Ax + b) \\ x - \Pi_C(x - z) \\ c - A^*y - z \end{pmatrix}, \quad \forall w = (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n. \tag{7}$$

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

$$h(\bar{y}^{k+1}, \bar{z}^{k+1}) := -\langle b, \bar{y}^{k+1}\rangle + \delta_C^*(-\bar{z}^{k+1}) + \langle b, y^*\rangle - \delta_C^*(-z^*), \quad \forall k \ge 0,$$

where (y^*, z^*) is the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k)\}$. Based on the iteration complexity presented in Proposition 3, we can derive the complexity result of the HPR method with semi-proximal terms in terms of the KKT residual and the objective error, as stated in the following theorem.

Theorem 1 (Theorem 3.7 in [42]) *Suppose that Assumption 1 holds. Let* $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ *be the sequence generated by the HPR method with semi-proximal terms in Algorithm 1, and let* $w^* = (y^*, z^*, x^*)$ *be the limit point of the sequence* $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ *and* $R_0 = \|w^0 - w^*\|_{\mathcal{M}}$. *Then for all* $k \geq 0$, *we have the following iteration complexity bounds:*

$$\|\mathcal{R}(\bar{w}^{k+1})\| \leq \left(\frac{\sigma(\|A\| + \|\sqrt{\mathcal{T}_1}\|) + 1}{\sqrt{\sigma}}\right) \frac{R_0}{(k+1)} \tag{8}$$

and

$$\left(\frac{-1}{\sqrt{\sigma}}\|x^*\|\right) \frac{R_0}{(k+1)} \leq h(\bar{y}^{k+1}, \bar{z}^{k+1}) \leq \left(3R_0 + \frac{1}{\sqrt{\sigma}}\|x^*\|\right) \frac{R_0}{(k+1)}. \tag{9}$$

Remark 1 There are several complexity results related to the KKT residual of ADMM-type methods for solving LP in the literature. In the ergodic sense, Monteiro and Svaiter [31] established an ergodic $O(1/k)$ iteration complexity for ADMM with a unit dual step size in terms of the ε-subdifferential, which, as mentioned in the introduction, implies an ergodic $O(1/\sqrt{k})$ iteration complexity for the KKT residual. Recently, building on the ergodic $O(1/k)$ iteration complexity of the PDHG established by Chambolle and Pock [9,10], Applegate et al. [3] derived an ergodic $O(1/k)$ iteration complexity of the PDHG for solving LP, measured by primal feasibility violation, dual feasibility violation, and the primal-dual gap, all of which can be inferred from the KKT residual. In the nonergodic sense, Cui et al. [11] showed that a majorized ADMM with semi-proximal terms achieves a nonergodic $O(1/\sqrt{k})$ iteration complexity for the KKT residual. Compared to these existing methods, the HPR method attains a much improved $O(1/k)$ iteration complexity in terms of the KKT residual as in (8), providing stronger theoretical guarantees that, in turn, support practical advantages in solving large-scale LP problems efficiently. For more complexity results, please refer to [42] and the references therein.

3 A Halpern Peaceman-Rachford solver for solving LP

In this section, we present an HPR-LP solver for solving large-scale LP problems, as outlined in Algorithm 2. In the HPR-LP solver, we integrate a restart strategy and adaptive updates of the penalty parameter σ into the HPR method with semi-proximal terms.

3.1 Restart criteria

From Theorem 1, we know that the base algorithm, the HPR method with semi-proximal terms, achieves the iteration complexity of $O(1/k)$ with respect to the KKT residual and the objective error. This result is derived from the

Algorithm 2 HPR-LP: A Halpern Peaceman-Rachford method for the problem (2)

Input: Let $\mathcal{T}_1 : \mathbb{R}^m \to \mathbb{R}^m$ be a self-adjoint positive semidefinite linear operator such that $\mathcal{T}_1 + AA^*$ is positive definite. Denote $w = (y, z, x)$ and $\bar{w} = (\bar{y}, \bar{z}, \bar{x})$. Choose an initial point $w^{0,0} = (y^{0,0}, z^{0,0}, x^{0,0}) \in D \times \mathbb{R}^n \times \mathbb{R}^n$.

Initialization: Set the outer loop counter $r = 0$, the total loop counter $k = 0$, and the initial penalty parameter $\sigma_0 > 0$.

repeat

  initialize the inner loop: set inner loop counter $t = 0$;

  repeat

    $\bar{z}^{r,t+1} = \arg\min_{z \in \mathbb{R}^n} \left\{ L_{\sigma_r}\left(y^{r,t}, z; x^{r,t}\right) \right\}$;

    $\bar{x}^{r,t+1} = x^{r,t} + \sigma_r(A^* y^{r,t} + \bar{z}^{r,t+1} - c)$;

    $\bar{y}^{r,t+1} = \arg\min_{y \in \mathbb{R}^m} \left\{ L_{\sigma_r}\left(y, \bar{z}^{r,t+1}; \bar{x}^{r,t+1}\right) + \frac{\sigma_r}{2}\|y - y^{r,t}\|^2_{\mathcal{T}_1} \right\}$;

    $\hat{w}^{r,t+1} = 2\bar{w}^{r,t+1} - w^{r,t}$;

    $w^{r,t+1} = \frac{1}{t+2} w^{r,0} + \frac{t+1}{t+2} \hat{w}^{r,t+1}$;

    $t = t + 1$, $k = k + 1$;

  until one of the restart criteria holds or termination criteria hold

  restart the inner loop: $\tau_r = t, w^{r+1,0} = \bar{w}^{r,\tau_r}$,

  $\sigma_{r+1} = \textbf{SigmaUpdate}(\bar{w}^{r,\tau_r}, w^{r,0}, \mathcal{T}_1, A)$, $r = r + 1$;

until termination criteria hold

Output: $\{\bar{w}^{r,t}\}$.

complexity analysis in Proposition 3, which offers a tighter bound for designing restart criteria. Thus, we define the following merit function:

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

where w^* is an arbitrary solution to the KKT system (3). Note that $R_{r,0}$ represents the upper bound (disregarding the factor 2) in the complexity results in Proposition 3 at the r-th outer loop. A natural strategy is to restart the inner loop when $R_{r,t}$ has sufficiently decreased compared to $R_{r,0}$, i.e., $R_{r,t} \leq \alpha_1 R_{r,0}$, where $\alpha_1 \in (0, 1)$. Unfortunately, in practice, if α_1 is too small, the algorithm is likely to fail to achieve this sufficient reduction. Hence, we also consider the length of the inner loop and the oscillation of the merit function $R_{r,t}$. These ideas have been implemented in PDLP with different merit functions [2, 28, 30].[6] In addition, since w^* is unknown, we approximate $R_{r,t}$ using the following expression, inspired by Proposition 3:

$$\widetilde{R}_{r,t} = \|w^{r,t} - \hat{w}^{r,t+1}\|_{\mathcal{M}}.$$

Consequently, the restart criteria in HPR-LP are defined as follows:

[6] In [2], Applegate et al. used a normalized duality gap as the merit function, whereas Lu et al. selected a weighted KKT residual as the merit function in [28,30].

1. Sufficient decay of $\widetilde{R}_{r,t+1}$:
$$\widetilde{R}_{r,t+1} \le \alpha_1 \widetilde{R}_{r,0}; \tag{10}$$
2. Necessary decay + no local progress of $\widetilde{R}_{r,t+1}$:
$$\widetilde{R}_{r,t+1} \le \alpha_2 \widetilde{R}_{r,0} \quad \text{and} \quad \widetilde{R}_{r,t+1} > \widetilde{R}_{r,t}; \tag{11}$$
3. Long inner loop:
$$t \ge \alpha_3 k, \tag{12}$$

where $\alpha_1 \in (0, \alpha_2)$, $\alpha_2 \in (0,1)$, and $\alpha_3 \in (0,1)$. Once any of the three restart criteria is met, we restart the inner loop for the $(r+1)$-th iteration, set $w^{r+1,0} = \bar{w}^{r,\tau_r}$, and update σ_{r+1}.

3.2 Update rule for σ

The update rule for σ in HPR-LP is also derived from the complexity results of the HPR method with semi-proximal terms in Proposition 3. Specifically, we update σ_{r+1} at the $(r+1)$-th restart for any $r \ge 0$ by solving the following optimization problem:

$$\sigma_{r+1} := \arg\min_{\sigma} \left\| w^{r+1,0} - w^* \right\|_{\mathcal{M}}^2, \tag{13}$$

where $\left\| w^{r+1,0} - w^* \right\|_{\mathcal{M}}$ represents the upper bound of the complexity results in Proposition 3 at the $(r+1)$-th outer loop. A smaller upper bound is expected to lead to a smaller $\|w^{r+1,t} - \hat{w}^{r+1,t+1}\|_{\mathcal{M}}$ for any $t \ge 0$, which further results in a smaller KKT residual $\|\mathcal{R}(\bar{w}^{r+1,t+1})\|$. Substituting the definition of $\mathcal{M}$ from (5) into (13), we derive

$$\begin{aligned}
\sigma_{r+1} &= \arg\min_{\sigma} \left\| w^{r+1,0} - w^* \right\|_{\mathcal{M}}^2 \\
&= \arg\min_{\sigma} \left(\sigma \|y^{r+1,0} - y^*\|_{\mathcal{T}_1}^2 + \sigma^{-1} \|x^{r+1,0} - x^* + \sigma A^*(y^{r+1,0} - y^*)\|^2 \right) \\
&= \sqrt{\frac{\|x^{r+1,0} - x^*\|^2}{\|y^{r+1,0} - y^*\|_{\mathcal{T}_1}^2 + \|A^*(y^{r+1,0} - y^*)\|^2}}.
\end{aligned} \tag{14}$$

Since computing $\|x^{r+1,0} - x^*\|$ and $\|y^{r+1,0} - y^*\|_{\mathcal{T}_1}^2 + \|A^*(y^{r+1,0} - y^*)\|^2$ are not implementable, we approximate these terms in HPR-LP using

$$\Delta_x := \|\bar{x}^{r,\tau_r} - x^{r,0}\| \ \text{ and } \ \Delta_y := \sqrt{\|\bar{y}^{r,\tau_r} - y^{r,0}\|_{\mathcal{T}_1}^2 + \|A^*(\bar{y}^{r,\tau_r} - y^{r,0})\|^2}, \tag{15}$$

respectively. Consequently, we update σ_{r+1} as follows:

$$\sigma_{r+1} = \frac{\Delta_x}{\Delta_y}. \tag{16}$$

Note that the approximations Δ_x and Δ_y may deviate significantly from the true values, so we update σ using formula (16) only when the following conditions are met; otherwise, we reset σ to 1:

1. Δ_x and Δ_y are within a suitable range:

$$\Delta_x \in (10^{-16}, 10^{12}), \quad \Delta_y \in (10^{-16}, 10^{12}); \tag{17}$$

2. The relative infeasibilities of primal and dual problems should not differ excessively:

$$\frac{\text{error}_d}{\text{error}_p} \in (10^{-8}, 10^{8}), \tag{18}$$

where

$$\text{error}_p := \frac{\|\Pi_D(b - A\bar{x}^{r,\tau_r})\|}{1 + \|b\|}, \quad \text{error}_d := \frac{\|c - A^*\bar{y}^{r,\tau_r} - \bar{z}^{r,\tau_r}\|}{1 + \|c\|}.$$

In summary, the update rule for σ is presented in Algorithm 3.

Algorithm 3 SigmaUpdate

Input: $(\bar{w}^{r,\tau_r}, w^{r,0}, \mathcal{T}_1, A)$.
Calculate Δ_x and Δ_y defined in (15);
if conditions (17) and (18) are satisfied **then**
    $\sigma_{r+1} = \dfrac{\Delta_x}{\Delta_y}$;
else
    $\sigma_{r+1} = 1$;
end if
Output: σ_{r+1}.

Remark 2 The update rule for σ is not only applicable to LP problems but can also be directly extended to the HPR method with semi-proximal terms [42] for solving more general convex optimization problems (27).

We discuss two special choices of proximal operator $\mathcal{T}_1$ to obtain the update formula of σ as follows. For other choices of $\mathcal{T}_1$, one can use (16) to determine σ.

1. $\mathcal{T}_1 = 0$. For example, consider the LP problems without inequality constraint (i.e., $m_2 = 0$). At the r-th outer loop and t-th inner loop, $\bar{y}^{r,t+1}$ can be obtained by solving the following linear equations:

$$AA^*\bar{y}^{r,t+1} = \frac{1}{\sigma}(b - A(\bar{x}^{r,t+1} + \sigma(\bar{z}^{r,t+1} - c))). \tag{19}$$

Solving the linear equations (19) by the direct method is affordable in many applications such as in the optimal transport problem [45] or the Wasserstein barycenter problem [46]. According to (16), if conditions (17) and (18) are satisfied, then σ_{r+1} is computed as:

$$\sigma_{r+1} = \frac{\|\bar{x}^{r,\tau_r} - x^{r,0}\|}{\|A^*(\bar{y}^{r,\tau_r} - y^{r,0})\|}. \tag{20}$$

2. $\mathcal{T}_1 = \lambda I_m - AA^*$ with $\lambda \geq \lambda_1(AA^*)$ as proposed in [14,9,43]. This applies when inequality constraints are present (i.e., $m_2 > 0$) or when solving the linear equations (19) directly is not affordable. In this case,

$$\begin{cases} \bar{y}_1^{r,t+1} = y_1^{r,t} + \dfrac{1}{\lambda}(b_1/\sigma - A_1 R_y), \\ \bar{y}_2^{r,t+1} = \Pi_{\mathbb{R}^{m_2}_+}\left(y_2^{r,t} + \frac{1}{\lambda}(b_2/\sigma - A_2 R_y)\right), \end{cases} \tag{21}$$

where $R_y := \bar{x}^{r,t+1}/\sigma + (A^* y^{r,t} + \bar{z}^{r,t+1} - c)$. If conditions (17) and (18) are satisfied, then σ_{r+1} is given by:

$$\sigma_{r+1} = \frac{1}{\sqrt{\lambda}} \frac{\|\bar{x}^{r,\tau_r} - x^{r,0}\|}{\|\bar{y}^{r,\tau_r} - y^{r,0}\|}. \tag{22}$$

Remark 3 The formula (22) is similar to the primal weight ω update formula with $\theta = 1$ in Algorithm 3 of [2], except for the inclusion of the term λ.

3.3 The GPU implementation of the HPR-LP

We first present the update formulas for each subproblem in HPR-LP (Steps 6-8). Specifically, for any $r \geq 0$ and $t \geq 0$, the update of $\bar{z}^{r,t+1}$ is given by:

$$\begin{aligned} \bar{z}^{r,t+1} &= \arg\min_{z \in \mathbb{R}^n} \left\{ L_{\sigma_r}\left(y^{r,t}, z; x^{r,t}\right) \right\} \\ &= \frac{1}{\sigma_r}\Big(\Pi_C\big(x^{r,t} + \sigma_r(A^* y^{r,t} - c)\big) - \big(x^{r,t} + \sigma_r(A^* y^{r,t} - c)\big) \Big). \end{aligned} \tag{23}$$

Next, the update of $\bar{x}^{r,t+1}$ is:

$$\bar{x}^{r,t+1} = x^{r,t} + \sigma_r(A^* y^{r,t} + \bar{z}^{r,t+1} - c) = \Pi_C\big(x^{r,t} + \sigma_r(A^* y^{r,t} - c)\big). \tag{24}$$

For general LP problems, we set $\mathcal{T}_1 = \lambda I_m - AA^*$ with $\lambda \geq \lambda_1(AA^*)$ in the HPR-LP. Consequently, following from (21) and (24), the update for $\bar{y}^{r,t+1}$ is given by:

$$\begin{cases} \bar{y}_1^{r,t+1} = y_1^{r,t} + \dfrac{1}{\lambda\sigma_r}\Big(b_1 - A_1(2\bar{x}^{r,t+1} - x^{r,t})\Big), \\ \bar{y}_2^{r,t+1} = \Pi_{\mathbb{R}^{m_2}_+}\Big(y_2^{r,t} + \dfrac{1}{\lambda\sigma_r}\Big(b_2 - A_2(2\bar{x}^{r,t+1} - x^{r,t})\Big)\Big). \end{cases} \tag{25}$$

By combining (23), (24), and (25), we observe that it is not necessary to compute $\bar{z}^{r,t+1}$ at every iteration. Instead, we only need to evaluate $\bar{z}^{r,t+1}$ using (23) when checking the termination criteria for stopping HPR-LP. Furthermore, the update of (23), (24), and (25) mainly involves the matrix-vector multiplications, vector additions, and projections. The overall per-iteration complexity of HPR-LP is $O(\text{nnz}(A))$, where $\text{nnz}(A)$ denotes the number of nonzero entries in A.

Moreover, to fully utilize the parallel computing power of GPUs, we implement custom CUDA kernels for (23), (24), and (25). For matrix-vector multiplications, we utilize `cusparseSpMV()` from the cuSPARSE library, which applies the `CUSPARSE_SPMV_CSR_ALG2` algorithm to ensure deterministic results.

4 Numerical experiments

In this section, we compare the performance of HPR-LP implemented in Julia [6] with cuPDLP.jl [28] on a GPU. The experimental setup is detailed in Section 4.1. Section 4.2 discusses the performance of these two solvers on Mittelmann's LP benchmark set.[7] Section 4.3 presents numerical results on LP relaxations of instances from the MIPLIB 2017 collection [17]. Finally, Section 4.4 highlights the numerical results on extremely large instances, including LPs generated from quadratic assignment problems (QAPs) [8], the "zib03" instance as discussed by Koch et al. [24], and the LP formulation of the PageRank problem [32].

4.1 Experimental setup

Benchmark datasets. We conduct extensive performance experiments on both classical benchmark sets and extremely large-scale instances. Specifically, we evaluate HPR-LP and cuPDLP.jl [28] on two classic benchmark datasets: Mittelmann's LP benchmark set and LP relaxations of instances from the MIPLIB 2017 collection. We use 49 publicly available Mittelmann's LP benchmark instances. For the MIPLIB 2017 set, we initially select 383 instances based on the criteria outlined in [28]. After excluding three instances during Gurobi's presolve [19], we retain 380 instances.[8] We then split the MIP relaxations into three classes based on the number of nonzeros in the constraint matrix, as shown in Table 1, following the approach in [28]. In addition, to further explore the limits of the capability of HPR-LP, we employ several extremely large-scale LP problems referenced in [30], including LPs generated from QAPs [8] the "zib03" instance from [24], and several instances of PageRank problem [32].

Table 1: Scales of instances in MIP relaxations.

	Small	Medium	Large
Number of nonzeros	100K-1M	1M-10M	>10M
Number of instances	268	94	18

Software and computing environment. HPR-LP is implemented in Julia [6], referred to as HPR-LP.jl. Similarly, cuPDLP.jl [28], the GPU version of PDLP, is also implemented in Julia.[9] All tested solvers are run on an NVIDIA

[7] https://plato.asu.edu/ftp/lpfeas.html.

[8] Two instances are identified as unbounded by Gurobi's presolve, and one instance is solved by Gurobi's presolve.

[9] We downloaded the cuPDLP.jl from https://github.com/jinwen-yang/cuPDLP.jl on July 24th, 2024. The infeasibility detection function of cuPDLP.jl is disabled.

A100-SXM4-80GB GPU with CUDA 12.3, and the experiments are conducted in Julia 1.10.4 on Ubuntu 22.04.3 LTS.

Presolve and preconditioning. To assess the impact of presolve on the FOMs, we compare all tested algorithms on two sets of instances: the original instances without presolve and the instances with presolve by Gurobi 11.0.3 (academic license). For preconditioning in HPR-LP, we perform 10 iterations of Ruiz scaling [37], followed by the bidiagonal preconditioning used by Pock and Chambolle [35] with $\alpha = 1$. Finally, we normalize b and c divided by $\|b\|+1$ and $\|c\|+1$, respectively. For cuPDLP.jl [28], we use the default preconditioning settings.

Initialization and parameter setting. We initialize HPR-LP with the origin point and set the penalty parameter $\sigma_0 = 1$. After preconditioning, we choose $\mathcal{T}_1 = \lambda_1(AA^*)I_m - AA^*$ and compute $\lambda_1(AA^*)$ using the power method [18]. In HPR-LP, the restart criteria are based on conditions (10), (11), and (12), with parameters $\alpha_1 = 0.2$, $\alpha_2 = 0.6$, and $\alpha_3 = 0.2$. The penalty parameter σ is updated using formula (22). For cuPDLP.jl [28], the default settings are applied.

Termination criteria. In HPR-LP, the sequence $\{\bar{w}^{r,t}\}$ is used to check the stopping criteria, which automatically satisfies $\bar{x}^{r,t} \in C$ and $\bar{y}^{r,t} \in D$ for any $r \geq 0$, and $t \geq 1$. We terminate HPR-LP when the following stopping criteria used in PDLP [2,28,30] are satisfied for the tolerance $\varepsilon \in (0, \infty)$:

$$
\begin{aligned}
&|\langle b, y\rangle - \delta_C^*(-z) - \langle c, x\rangle| \leq \varepsilon \left(1 + |\langle b, y\rangle - \delta_C^*(-z)| + |\langle c, x\rangle|\right), \\
&\|\Pi_D(b - Ax)\| \leq \varepsilon \left(1 + \|b\|\right), \\
&\|c - A^*y - z\| \leq \varepsilon \left(1 + \|c\|\right).
\end{aligned}
$$

We evaluate the performance of all tested algorithms with $\varepsilon = 10^{-4}$, 10^{-6}, and 10^{-8} for all the datasets. Moreover, in HPR-LP.jl, we check the termination and restart criteria every 150 iterations, whereas cuPDLP.jl uses its default setting of checking termination criteria every 64 iterations.

Time limit. In Section 4.3, we impose 15000 seconds as the time limit in Mittelmann's benchmark. For Section 4.2, we impose a time limit of 3600 seconds on instances with small-sized and medium-sized instances, and a time limit of 18000 seconds for large instances. For Section 4.4, we impose 18000 seconds as the time limit for the LP instances generated by QAPs and PageRank problems, and 36000 seconds for the "zib03" instance.

Shifted geometric mean. We use the shifted geometric mean of solving time, as employed in Mittelmann's benchmarks, to measure the performance of solvers on a collection of problems. Specifically, the shifted geometric mean

is defined as

$$(\prod_{i=1}^{n}(t_i+\Delta))^{1/n}-\Delta,$$

where t_i is the solving time in seconds for the i-th instance. We shift by $\Delta = 10$ and denote this measure as SGM10. If an instance remains unsolved, the solving time is set to the corresponding time limit. As with cuPDLP.jl [28], we exclude the time required for reading data, presolving, and preconditioning. For cuPDLP.jl, we use the default timing settings. For HPR-LP.jl, in addition to the algorithm runtime, we also include the time spent on the power method.

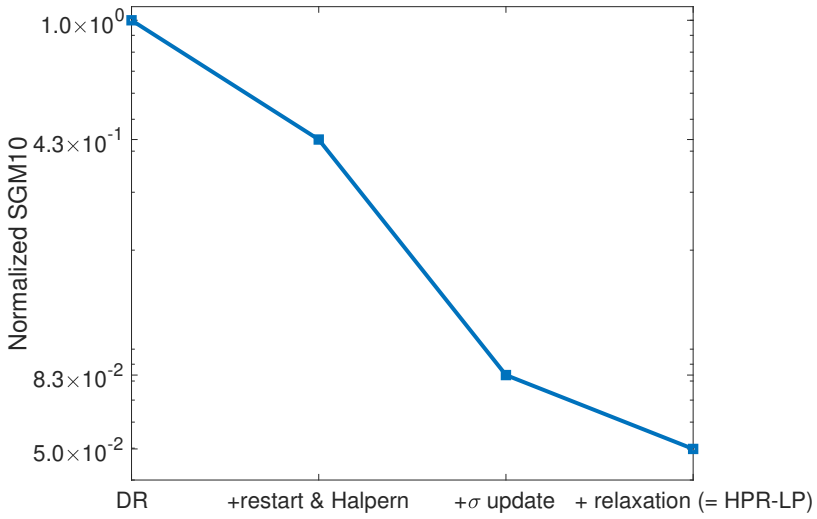
4.2 Mittelmann's LP benchmark set

Mittelmann's LP benchmark set is a standard benchmark for evaluating the numerical performance of different LP solvers. In this experiment, we first analyze the impact of the restart strategy, the Halpern iteration, the update rule for σ, and the relaxation step (Step 9 in Algorithm 2). We then compare HPR-LP.jl with cuPDLP.jl on Mittelmann's LP benchmark set.

Figure 1 summarizes the relative impact of HPR-LP's enhancements on 49 instances of Mittelmann's LP benchmark set without presolve at a tolerance of 10^{-8}. As shown in subtable (a) of Figure 1, the enhancements described in Section 3 significantly improve the performance of the baseline Douglas-Rachford (DR) method, which corresponds to Steps 1–3 in Algorithm 1. Meanwhile, subfigure (b) of Figure 1 presents the normalized SGM10 values, computed relative to the baseline DR method, offering a clear visual representation of performance gains. Specifically, incorporating restart criteria and the Halpern iteration, the Halpern DR method for LP (HDR-LP) with a fixed σ solves 4 more problems than the DR method and achieves a **2.30**× speedup in terms of SGM10. Furthermore, introducing the adaptive update rule for σ allows HDR-LP to solve 9 more problems than HDR-LP with a fixed σ, resulting in a **5.20**× speedup in terms of SGM10. Finally, by incorporating an additional relaxation step (Step 9 in Algorithm 2), HPR-LP benefits from a larger step size than HDR-LP, leading to a **1.67**× speedup over HDR-LP in terms of SGM10.

(a) SGM10 and the number of solved problems.

Enhancements	Solved	SGM10
DR	27	2087.3
+ restart & Halpern (=HDR-LP with fixed σ)	31	905.7
+ σ update (=HDR-LP)	42	174.1
+ relaxation (=HPR-LP)	43	103.8



(b) Normalized SGM10.

Fig. 1: Relative impact of HPR-LP's enhancements on Mittelmann's LP benchmark set without presolve.

The numerical performance of HPR-LP.jl and cuPDLP.jl, both with and without presolve, is presented in Tables 2 and 3, respectively. The main results are summarized as follows:

- HPR-LP.jl consistently solves **3** to **5** more problems than cuPDLP.jl does across all tolerance levels, as shown in Tables 2 and 3.
- In terms of SGM10, HPR-LP.jl outperforms cuPDLP.jl on Mittelmann's LP benchmark set across all tolerance levels. For instance, as shown in Table 2, HPR-LP.jl achieves a **3.71x** speedup over cuPDLP.jl to obtain a solution with a 10^{-8} relative accuracy for the presolved dataset. Similarly, in Table 3, HPR-LP.jl achieves a **2.68x** speedup over cuPDLP.jl to attain the same tolerance for the unpresolved dataset.

- HPR-LP.jl exhibits better performance on the presolved dataset compared to the unpresolved dataset, since HPR-LP.jl solves more problems on the presolved Mittelmann's LP benchmark set across all tolerance levels.

Table 2: Numerical performance of different solvers on 49 instances of Mittelmann's LP benchmark set with presolve.

Tolerance	10^{-4}		10^{-6}		10^{-8}	
Solvers	SGM10	Solved	SGM10	Solved	SGM10	Solved
cuPDLP.jl	60.0	46	118.6	45	220.6	43
HPR-LP.jl	17.4	49	31.8	49	59.4	48

Table 3: Numerical performance of different solvers on 49 instances of Mittelmann's LP benchmark set without presolve.

Tolerance	10^{-4}		10^{-6}		10^{-8}	
Solvers	SGM10	Solved	SGM10	Solved	SGM10	Solved
cuPDLP.jl	76.9	42	156.2	41	277.9	40
HPR-LP.jl	30.2	47	69.1	44	103.8	43

To further compare HPR-LP.jl with cuPDLP.jl, we present the shifted geometric mean of iteration counts for different solvers on 49 instances of Mittelmann's LP benchmark set in Table 4, similar to the SGM10 metric for time. As shown in Table 4, HPR-LP.jl requires fewer iterations than cuPDLP.jl across different tolerance levels. Additionally, Table 5 reports the per-iteration time of both solvers. On average, the per-iteration time of cuPDLP.jl is **2.59**$\times$ that of HPR-LP.jl with presolve and **2.22**$\times$ without presolve. One possible reason is that cuPDLP.jl employs a line search strategy to achieve a larger step size, which incurs significant computational overhead [28]. In contrast, HPR-LP.jl makes use of the theoretical advance in [42] to choose the desirable relaxation parameter value of 2 (Step 9 in Algorithm 2) to obtain a large step size, providing a guaranteed theoretical foundation for simpler and more efficient implementation.

Table 4: Shifted geometric mean of iteration counts for different solvers on 49 instances of Mittelmann's LP benchmark set, with and without presolve.

	With Presolve			Without Presolve		
Solver	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}
cuPDLP.jl	47694.3	120748.1	253827.3	42997.0	125972.9	260003.4
HPR-LP.jl	27661.1	71380.7	150375.0	36018.2	115121.2	206069.4

Table 5: Per-iteration time in seconds of different solves for Mittelmann's LP benchmark set.

	With Presolve		Without Presolve	
Metric	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl
Median	1.4e-4	4.3e-4	2.0e-4	5.2e-4
Mean	3.7e-4	9.6e-4	5.4e-4	1.2e-3
Standard Deviation	9.0e-4	1.7e-3	7.2e-5	3.1e-4
Min	7.7e-5	3.0e-4	6.2e-3	1.1e-2
Max	6.2e-3	1.1e-2	1.1e-3	2.0e-3

4.3 MIP relaxations

In this experiment, we compare HPR-LP.jl with cuPDLP.jl on the MIP relaxations set, which includes 380 instances. The numerical performance of these two solvers, both with and without presolve, is presented in Tables 6 and 7, respectively. The main results are summarized as follows:

- With a 10^{-8} accuracy, HPR-LP.jl solves **7** more problems than cuPDLP.jl does across the 380 presolved MIP relaxation instances, as shown in Table 6. Moreover, on the unpresolved dataset, HPR-LP.jl solves more problems than cuPDLP.jl does in different stopping tolerances, as shown in Table 7.
- In terms of SGM10 across 380 instances, HPR-LP.jl consistently outperforms cuPDLP.jl at all tolerance levels. As shown in Table 6, HPR-LP.jl achieves a **2.39x** speedup over cuPDLP.jl to obtain a solution with a 10^{-8} accuracy for the presolved dataset. Similarly, in Table 7, HPR-LP.jl achieves a **2.03x** speedup over cuPDLP.jl to obtain a solution with the same accuracy for the unpresolved dataset.

Table 6: Numerical performance of different solvers on 380 instances of MIP relaxations with presolve.

Classes	Tolerance	10^{-4}		10^{-6}		10^{-8}	
	Solvers	SGM10	Solved	SGM10	Solved	SGM10	Solved
Small (268)	cuPDLP.jl	8.4	262	15.3	261	23.7	254
	HPR-LP.jl	4.1	262	6.3	261	8.9	261
Medium (94)	cuPDLP.jl	10.4	94	23.6	92	35.4	92
	HPR-LP.jl	5.8	94	11.0	92	15.8	92
Large (18)	cuPDLP.jl	31.7	17	64.1	17	102.4	17
	HPR-LP.jl	22.9	17	44.0	17	76.0	17
Total (380)	cuPDLP.jl	9.6	373	18.6	370	28.4	363
	HPR-LP.jl	5.1	373	8.3	370	11.9	370

Table 7: Numerical performance of different solvers on 380 instances of MIP relaxations without presolve.

Classes	Tolerance	10^{-4}		10^{-6}		10^{-8}	
	Solvers	SGM10	Solved	SGM10	Solved	SGM10	Solved
Small (268)	cuPDLP.jl	11.5	266	20.0	262	27.7	258
	HPR-LP.jl	4.6	267	7.6	265	12.6	259
Medium (94)	cuPDLP.jl	14.9	90	26.7	89	43.8	87
	HPR-LP.jl	7.4	92	13.7	91	19.8	90
Large (18)	cuPDLP.jl	129.8	16	253.3	15	442.2	14
	HPR-LP.jl	117.6	17	260.7	15	428.6	14
Total (380)	cuPDLP.jl	14.3	372	25.0	366	36.3	359
	HPR-LP.jl	6.9	376	11.6	371	17.9	363

4.4 Large-scale applications: QAP, ZIB problem, and PageRank instances

In this experiment, we evaluate the performance of HPR-LP.jl and cuPDLP.jl on extremely large-scale LP datasets. Specifically, we first apply Adams-Johnson linearization [1] to generate LP instances of quadratic assignment problems

(QAPs) from QAPLIB [8], formulated as follows:

$$
\begin{aligned}
\min_{x,s} \quad & \sum_{i,j}\sum_{k,l} a_{ik}b_{jl}s_{ijkl} \\
\text{s.t.} \quad & \sum_{i} s_{ijkl} = x_{kl}, && j,k,l = 1,\dots,N, \\
& \sum_{j} s_{ijkl} = x_{kl}, && i,k,l = 1,\dots,N, \\
& s_{ijkl} = s_{klij}, && i,j,k,l = 1,\dots,N, \\
& s_{ijkl} \geq 0, && i,j,k,l = 1,\dots,N, \\
& \sum_{j} x_{ij} = 1, && i = 1,\dots,N, \\
& \sum_{i} x_{ij} = 1, && j = 1,\dots,N, \\
& 0 \leq x_{ij} \leq 1, && i,j = 1,\dots,N,
\end{aligned}
\tag{26}
$$

where $(a_{ik})_{N\times N}$ represents the flow matrix, and $(b_{jl})_{N\times N}$ represents the distance matrix in the facility location application. The solving time and SGM10 for the two tested algorithms on 20 QAP instances, both with and without presolve, are presented in Tables 8 and 9, respectively. Across all tolerance levels, HPR-LP.jl consistently outperforms cuPDLP.jl in terms of SGM10. For example, HPR-LP.jl achieves a **5.70x** speedup over cuPDLP.jl on the presolved dataset in terms of SGM10 to obtain a solution with a 10^{-8} relative accuracy, as shown in Table 8. On the unpresolved dataset, as shown in Table 9, HPR-LP.jl achieves a **2.57x** speedup over cuPDLP.jl in terms of SGM10 for the same relative accuracy.

Table 8: Solving time in seconds and SGM10 for different solvers on 20 QAP instances [8] with presolve.

Tolerance	10^{-4}		10^{-6}		10^{-8}	
Solver	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl
esc64a	5.8	5.4	6.2	7.0	6.9	14.6
lipa40a	0.6	5.5	3.5	35.2	19.3	240.2
lipa40b	0.5	3.2	2.5	11.4	13.7	93.8
lipa50a	6.2	11.2	8.2	72.7	38.5	365.3
lipa50b	0.9	7.0	5.8	20.3	32.6	172.1
lipa60a	2.5	20.8	12.4	141.8	85.5	1348.7
lipa60b	2.1	8.3	10.0	37.2	71.4	275.9
lipa70a	4.5	40.2	25.5	304.7	168.5	2111.9
lipa70b	4.8	15.0	22.0	76.0	129.3	571.2
sko56	1.8	18.0	8.1	108.5	100.0	550.0
sko64	3.1	28.6	12.3	183.4	125.0	1650.6
tai40a	0.5	3.8	1.9	12.9	14.3	99.9
tai40b	4.3	11.0	12.9	141.6	215.3	456.3
tai50a	1.0	6.4	4.2	24.4	31.4	186.8
tai50b	1.8	14.6	12.8	522.2	303.2	839.4
tai60a	2.7	9.1	8.4	46.6	66.1	344.3
tai60b	18.9	160.6	48.9	311.5	1099.8	7641.9
tai64c	2.1	4.9	2.1	5.6	2.6	5.5
tho40	0.5	7.7	1.8	44.3	14.5	360.1
wil50	1.3	5.9	4.1	34.3	56.1	341.2
SGM10	2.9	12.7	8.8	60.0	60.2	343.1

Table 9: Solving time in seconds and SGM10 for different solvers on 20 QAP instances [8] without presolve.

Tolerance	10^{-4}		10^{-6}		10^{-8}	
Solver	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl
esc64a	7.6	7.8	8.6	9.0	10.0	13.9
lipa40a	4.2	13.7	37.7	112.7	322.6	1601.5
lipa40b	5.1	16.6	45.9	110.6	421.5	1317.4
lipa50a	14.6	36.8	103.2	285.8	756.4	2446.9
lipa50b	14.7	38.9	115.2	372.0	1187.3	4244.2
lipa60a	27.2	95.5	218.2	479.7	1961.1	4556.0
lipa60b	36.0	68.8	273.9	823.0	2642.0	7991.0
lipa70a	54.4	170.0	461.0	1115.0	3639.9	10585.7
lipa70b	75.9	154.5	534.7	1383.3	5287.8	15409.4
sko56	32.6	55.7	515.6	1123.7	8511.9	15220.3
sko64	37.5	85.1	357.0	994.5	4125.7	9047.3
tai40a	5.8	12.7	45.2	163.0	426.9	1342.8
tai40b	8.5	32.8	184.9	361.2	3898.6	13098.4
tai50a	15.3	29.0	126.0	283.6	995.9	3322.1
tai50b	16.4	62.5	449.5	836.4	8287.1	14482.2
tai60a	35.3	76.8	272.5	811.0	2408.6	8713.1
tai60b	62.6	163.6	1939.7	1663.0	18000.0	18000.0
tai64c	4.8	9.2	7.7	12.6	12.4	35.5
tho40	5.0	15.2	39.4	141.7	613.3	1405.6
wil50	16.4	43.7	170.8	491.2	1171.7	3581.3
SGM10	18.9	43.9	150.7	342.4	1246.4	3202.5

The numerical results of the instance "zib03" in Table 10 show that HPR-LP.jl achieves a **4.47x** speedup over cuPDLP.jl on the presolved dataset and a **4.06x** speedup on the unpresolved dataset, both in terms of SGM10, to return a solution with a 10^{-8} relative accuracy. Moreover, we observe that for a relative accuracy of 10^{-4}, both HPR-LP.jl and cuPDLP.jl require more solution time with presolve than without. One possible explanation is that "zib03" has been preprocessed by CPLEX 11, and excessive preprocessing may not always enhance algorithm performance. In some cases, it can even be detrimental by introducing numerical issues and disrupting useful problem structures. For instance, in this case, the coefficients of c originally range from $[1, 2]$, but after presolve, they shift to [1e-6, 3], potentially affecting numerical stability.

Table 10: Solving time in seconds for the "zib03" instance [24].

Tolerance	10^{-4}		10^{-6}		10^{-8}	
Solver	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl
With presolve[a]	273.8	351.9	1317.2	1634.6	3685.8	16462.2
Without presolve[b]	154.2	237.7	1063.6	1963.9	4865.3	19746.4

[a] Although "zib03" has been preprocessed by CPLEX 11, we still apply Gurobi's presolve function to this data for consistency in the experimental setup. The matrix A in "zib03" contains 19,701,908 rows, 29,069,187 columns, and 104,300,584 non-zeros after presolve.
[b] The matrix A in "zib03" contains 19,731,970 rows, 29,128,799 columns, and 104,422,573 non-zeros without presolve.
[c] The commercial LP solver COPT used 16.5 hours to solve this instance on an AMD Ryzen 9 5900X [30].

Following the approach in [2], we generate multiple instances of the PageRank problem with varying numbers of nodes. In this experiment, the presolve procedure has minimal impact on these instances. Therefore, we focus on reporting the numerical results for all tested solvers on the original (unpresolved) dataset, as summarized in Table 11. Across all tolerance levels, HPR-LP.jl consistently outperforms cuPDLP.jl in terms of SGM10. For instance, on the unpresolved dataset, HPR-LP.jl achieves a **2.16**× speedup over cuPDLP.jl when obtaining a solution with 10^{-8} relative accuracy, as shown in Table 11.

Table 11: Solving time in seconds and SGM10 for different solvers on 4 PageRank instances without presolve.

Tolerance	10^{-4}		10^{-6}		10^{-8}	
Nodes	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl	HPR-LP.jl	cuPDLP.jl
10^4	0.1	1.4	0.1	1.5	0.1	1.6
10^5	0.1	1.4	0.1	1.5	0.2	1.8
10^6	0.4	1.7	0.5	2.4	0.8	2.8
10^7	9.9	7.5	15.2	32.6	20.7	46.6
SGM10	2.0	2.8	2.8	6.3	3.6	7.8

5 Conclusion

In this paper, we developed an HPR-LP solver for solving large-scale LP problems, which integrates an adaptive strategy of restart and penalty parameter update into an HPR method with semi-proximal terms. Extensive numerical results on LP benchmark datasets showcased the efficiency of the HPR-LP solver in computing solutions varying from low to high accuracy. Nevertheless, there are still a few instances where the HPR-LP solver fails to obtain

solutions within the allocated time. Exploring an accelerated sPADMM with a faster iteration complexity than $O(1/k)$ in terms of the KKT residual may be beneficial. One possible way to improve the performance of the HPR-LP solver is to incorporate the fast Krasnosel'skii-Mann iteration, which can yield an iteration complexity of $o(1/k)$ in terms of the KKT residual and the objective error [42]. Another possible way is to design a similar algorithm with a linear rate better than that possessed by sPADMM [21]. We leave these as our future research directions.

A The equivalence between the Halpern accelerating pADMM without proximal terms and the HPR method without proximal terms

Let $\mathbb{X}$, $\mathbb{Y}$, and $\mathbb{Z}$ be three finite-dimensional real Euclidean spaces, each equipped with an inner product $\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$. In this appendix, we aim to establish the equivalence between the HPR method without proximal terms [46] and the Halpern accelerating pADMM without proximal terms [42] for solving the following convex optimization problem:

$$\begin{array}{ll}\min_{y\in\mathbb{Y},z\in\mathbb{Z}} & f_1(y)+f_2(z)\\ \text{subject to} & B_1y+B_2z=c,\end{array} \tag{27}$$

where $f_1:\mathbb{Y}\to(-\infty,+\infty]$ and $f_2:\mathbb{Z}\to(-\infty,+\infty]$ are two proper closed convex functions, $B_1:\mathbb{Y}\to\mathbb{X}$ and $B_2:\mathbb{Z}\to\mathbb{X}$ are two given linear operators, and $c\in\mathbb{X}$ is a given point. Let $\sigma>0$ be a given penalty parameter. The augmented Lagrangian function of problem (27) is defined by, for any $(y,z,x)\in\mathbb{Y}\times\mathbb{Z}\times\mathbb{X}$,

$$L_\sigma(y,z;x):=f_1(y)+f_2(z)+\langle x,B_1y+B_2z-c\rangle+\frac{\sigma}{2}\|B_1y+B_2z-c\|^2.$$

The dual of problem (27) is given by

$$\max_{x\in\mathbb{X}}\left\{-f_1^*(-B_1^*x)-f_2^*(-B_2^*x)-\langle c,x\rangle\right\}, \tag{28}$$

where $B_1^*:\mathbb{X}\to\mathbb{Y}$ and $B_2^*:\mathbb{X}\to\mathbb{Z}$ are the adjoint of B_1 and B_2, respectively. Subsequently, the HPR method without proximal terms [46] and the Halpern accelerating pADMM without proximal terms [42] for solving problem (27) are detailed in Algorithm 4 and Algorithm 5, respectively.

Proposition 4 *Assume that $\partial f_1(\cdot)+\sigma B_1^*B_1$ and $\partial f_2(\cdot)+\sigma B_2^*B_2$ are maximal and strongly monotone, respectively. Let $w^0=(y^0,z^0,x^0)\in\mathrm{dom}(f_1)\times\mathrm{dom}(f_2)\times\mathbb{X}$. Then, the sequence $\{(y^{k+1},z^{k+1},x^{k+\frac{1}{2}})\}$ generated by the HPR method without proximal terms in Algorithm 4, starting from the same initial point $(y^0,x^0)\in\mathrm{dom}(f_1)\times\mathbb{X}$, is equivalent to the sequence $\{(\bar{y}^{k+1},\bar{z}^{k+1},\bar{x}^{k+1})\}$ produced by the Halpern accelerating pADMM without proximal terms in Algorithm 5.*

Proof Based on the assumptions that $\partial f_1(\cdot)+\sigma B_1^*B_1$ and $\partial f_2(\cdot)+\sigma B_2^*B_2$ are both maximal and strongly monotone, we can conclude from [5, Corollary 11.17] that the solution to each subproblem in both algorithms exists and is unique. We then prove the statement in the proposition using the method of induction. For $k=0$, due to the same initial point (y^0,x^0), it is clear that:

$$z^1=\bar{z}^1,\quad x^{\frac{1}{2}}=\bar{x}^1,\quad\text{and}\quad y^1=\bar{y}^1. \tag{29}$$

Moreover, according to Steps 2, 4, and 5 in Algorithm 4, we know that $\tilde{x}^1$ satisfies:

$$\begin{aligned}\tilde{x}^1&=\tfrac{1}{2}(\tilde{x}^0+x^1)+\tfrac{\sigma}{2}B_1(y^0-y^1)\\&=\tfrac{1}{2}(\tilde{x}^0+x^{\frac{1}{2}}+\sigma(B_1y^1+B_2z^1-c))+\tfrac{\sigma}{2}B_1(y^0-y^1)\\&=\tfrac{1}{2}(x^{\frac{1}{2}}+\tilde{x}^0+\sigma(B_1y^0+B_2z^1-c))\\&=x^{\frac{1}{2}}.\end{aligned} \tag{30}$$

Algorithm 4 An HPR method without proximal terms [46] for solving convex optimization problem (27)

Input: $y^0 \in \mathrm{dom}(f_1)$, $x^0 \in \mathbb{X}$, and $\sigma > 0$.
Initialization: $\tilde{x}^0 := x^0$.
for $k = 0, 1, ...,$ **do**
  Step 1. $z^{k+1} = \arg\min_{z\in\mathbb{Z}} \left\{ L_\sigma(y^k, z; \tilde{x}^k) \right\}$;
  Step 2. $x^{k+\frac{1}{2}} = \tilde{x}^k + \sigma(B_1 y^k + B_2 z^{k+1} - c)$;
  Step 3. $y^{k+1} = \arg\min_{y\in\mathbb{Y}} \left\{ L_\sigma(y, z^{k+1}; x^{k+\frac{1}{2}}) \right\}$;
  Step 4. $x^{k+1} = x^{k+\frac{1}{2}} + \sigma(B_1 y^{k+1} + B_2 z^{k+1} - c)$;
  Step 5. $\tilde{x}^{k+1} = \left(\frac{1}{k+2}\tilde{x}^0 + \frac{k+1}{k+2} x^{k+1} \right) + \frac{\sigma}{k+2} \left[B_1 y^0 - B_1 y^{k+1} \right]$;
end for

Algorithm 5 A Halpern accelerating pADMM without proximal terms [42] for solving convex optimization problem (27)

Input: $w^0 = (y^0, z^0, x^0) \in \mathrm{dom}(f_1) \times \mathrm{dom}(f_2) \times \mathbb{X}$, and $\sigma > 0$. Denote $w = (y, z, x)$ and $\bar{w} = (\bar{y}, \bar{z}, \bar{x})$.
for $k = 0, 1, ...,$ **do**
  Step 1. $\bar{z}^{k+1} = \arg\min_{z\in\mathbb{Z}} \left\{ L_\sigma \left(y^k, z; x^k \right) \right\}$;
  Step 2. $\bar{x}^{k+1} = x^k + \sigma(B_1 y^k + B_2 \bar{z}^{k+1} - c)$;
  Step 3. $\bar{y}^{k+1} = \arg\min_{y\in\mathbb{Y}} \left\{ L_\sigma \left(y, \bar{z}^{k+1}; \bar{x}^{k+1} \right) \right\}$;
  Step 4. $\hat{w}^{k+1} = 2\bar{w}^{k+1} - w^k$;
  Step 5. $w^{k+1} = \frac{1}{k+2} w^0 + \frac{k+1}{k+2} \hat{w}^{k+1}$;
end for

It follows from Step 1 in Algorithm 4 that z^2 satisfies the following optimality condition:

$$\begin{aligned} & 0 \in \partial f_2(z^2) + B_2^* \tilde{x}^1 + \sigma B_2^*(B_1 y^1 + B_2 z^2 - c) \\ \Longleftrightarrow \ & 0 \in \partial f_2(z^2) + B_2^*(x^{\frac{1}{2}} + \sigma(B_1 y^1 + B_2 z^2 - c)). \end{aligned} \tag{31}$$

On the other hand, $\bar{z}^2$ obtained by Step 1 in Algorithm 5 should satisfy the following optimality condition:

$$0 \in \partial f_2(\bar{z}^2) + B_2^*(x^1 + \sigma(B_1 y^1 + B_2 \bar{z}^2 - c)). \tag{32}$$

Substituting

$$x^1 = \frac{1}{2} x^0 + \frac{1}{2}(2\bar{x}^1 - x^0) \quad \text{and} \quad y^1 = \frac{1}{2} y^0 + \frac{1}{2}(2\bar{y}^1 - y^0) \tag{33}$$

from Steps 4 and 5 in Algorithm 5 into (32), we obtain:

$$0 \in \partial f_2(\bar{z}^2) + B_2^*(\bar{x}^1 + \sigma(B_1 \bar{y}^1 + B_2 \bar{z}^2 - c)).$$

This, together with (29) and (31), implies:

$$z^2 = \bar{z}^2. \tag{34}$$

Furthermore, according to Step 2 in Algorithm 4 and (30), we know that:

$$x^{\frac{3}{2}} = \tilde{x}^1 + \sigma(B_1 y^1 + B_2 z^2 - c) = x^{\frac{1}{2}} + \sigma(B_1 y^1 + B_2 z^2 - c). \tag{35}$$

Also, from Step 2 in Algorithm 5 and (33), we know that $\bar{x}^2$ satisfies:

$$\begin{aligned}\bar{x}^2 &= x^1 + \sigma(B_1 y^1 + B_2 \bar{z}^2 - c)\\ &= \bar{x}^1 + \sigma(B_1 \bar{y}^1 + B_2 \bar{z}^2 - c).\end{aligned}$$

It follows from (29), (34), and (35) that:

$$x^{\frac{3}{2}} = \bar{x}^2, \tag{36}$$

which, together with (34), implies:

$$y^2 = \bar{y}^2 \tag{37}$$

by comparing Step 3 in Algorithms 4 and 5. Hence, the statement is true for $k = 1$.

Now, assume that the statement holds for some $k \geq 1$. Then, by the assumption of induction, we have

$$y^{k+1} = \bar{y}^{k+1}, \quad z^{k+1} = \bar{z}^{k+1}, \text{ and } x^{k+\frac{1}{2}} = \bar{x}^{k+1}. \tag{38}$$

According to Step 5 in Algorithm 4 and (38), we know that $\tilde{x}^{k+1}$ satisfies

$$\begin{aligned}\tilde{x}^{k+1} &= \left(\tfrac{1}{k+2}\tilde{x}^0 + \tfrac{k+1}{k+2}x^{k+1}\right) + \tfrac{\sigma}{k+2}\left[B_1 y^0 - B_1 y^{k+1}\right]\\ &= \tfrac{1}{k+2}\tilde{x}^0 + \tfrac{k+1}{k+2}(x^{k+1/2} + \sigma(B_1 y^{k+1} + B_2 z^{k+1} - c)) + \tfrac{\sigma}{k+2}B_1(y^0 - y^{k+1})\\ &= \tfrac{1}{k+2}x^0 + \tfrac{k+1}{k+2}(\bar{x}^{k+1} + \sigma(B_1 \bar{y}^{k+1} + B_2 \bar{z}^{k+1} - c)) + \tfrac{\sigma}{k+2}B_1(y^0 - \bar{y}^{k+1}).\end{aligned} \tag{39}$$

In addition, according to Step 1 in Algorithm 4, we know that z^{k+2} satisfies

$$0 \in \partial f_2(z^{k+2}) + B_2^* \tilde{x}^{k+1} + \sigma B_2^*(B_1 y^{k+1} + B_2 z^{k+2} - c). \tag{40}$$

On the other hand, from Steps 2 and 5 in Algorithm 5, we know that x^{k+1} satisfies

$$\begin{aligned}x^{k+1} &= \tfrac{1}{k+2}x^0 + \tfrac{k+1}{k+2}(2\bar{x}^{k+1} - x^k)\\ &= \tfrac{1}{k+2}x^0 + \tfrac{k+1}{k+2}(\bar{x}^{k+1} + \sigma(B_1 y^k + B_2 \bar{z}^{k+1} - c))\end{aligned} \tag{41}$$

and y^{k+1} satisfies

$$y^{k+1} = \tfrac{1}{k+2}y^0 + \tfrac{k+1}{k+2}(2\bar{y}^{k+1} - y^k). \tag{42}$$

In addition, from Step 1 in Algorithm 5, we have

$$0 \in \partial f_2(\bar{z}^{k+2}) + B_2^*(x^{k+1} + \sigma\left(B_1 y^{k+1} + B_2 \bar{z}^{k+2} - c)\right). \tag{43}$$

Substituting (41) and (42) into (43), we can obtain

$$\begin{aligned}0 \in \partial f_2(\bar{z}^{k+2}) &+ B_2^*\left(\tfrac{1}{k+2}x^0 + \tfrac{k+1}{k+2}(\bar{x}^{k+1} + \sigma(B_1 \bar{y}^{k+1} + B_2 \bar{z}^{k+1} - c))\right)\\ &+ B_2^*\left(\tfrac{\sigma}{k+2}B_1(y^0 - \bar{y}^{k+1}) + \sigma(B_1 \bar{y}^{k+1} + B_2 \bar{z}^{k+2} - c)\right),\end{aligned}$$

which together with (39) implies

$$0 \in \partial f_2(\bar{z}^{k+2}) + B_2^*\left(\tilde{x}^{k+1} + \sigma(B_1 \bar{y}^{k+1} + B_2 \bar{z}^{k+2} - c)\right).$$

It follows from (38) and (40) that

$$z^{k+2} = \bar{z}^{k+2}. \tag{44}$$

Moreover, according to Step 2 in Algorithm 4, we know that $x^{k+\frac{3}{2}}$ satisfies

$$x^{k+\frac{3}{2}} = \tilde{x}^{k+1} + \sigma(B_1 y^{k+1} + B_2 z^{k+2} - c). \tag{45}$$

Also, from Step 2 in Algorithm 5, (39), (41) and (42), we know that $\bar{x}^{k+2}$ satisfies

$$\begin{aligned}\bar{x}^{k+2} &= x^{k+1} + \sigma(B_1 y^{k+1} + B_2 \bar{z}^{k+2} - c)\\ &= \tilde{x}^{k+1} + \sigma(B_1 \bar{y}^{k+1} + B_2 \bar{z}^{k+2} - c).\end{aligned}$$

It follows from (38), (44), and (45) that

$$x^{k+\frac{3}{2}} = \bar{x}^{k+2}. \tag{46}$$

Since $(\bar{z}^{k+2}, \bar{x}^{k+2}) = (z^{k+2}, x^{k+\frac{3}{2}})$, then we have

$$y^{k+2} = \bar{y}^{k+2} \tag{47}$$

by comparing Step 3 in Algorithms 4 and 5. Hence, from (44), (46), and (47), we know that the statement also holds for $k+1$. Thus, by induction, we have completed the proof. □

References

1. Adams, W.P., Johnson, T.A.: Improved linear programming based lower bounds for the quadratic assignment problem. In: P.M. Pardalos, H. Wolkowicz (eds.) Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993. American Mathematical Society (1994)
2. Applegate, D., Díaz, M., Hinder, O., Lu, H., Lubin, M., O'Donoghue, B., Schudy, W.: Practical large-scale linear programming using primal-dual hybrid gradient. In: Advances in Neural Information Processing System, vol. 34, pp. 20243–20257 (2021)
3. Applegate, D., Hinder, O., Lu, H., Lubin, M.: Faster first-order primal-dual methods for linear programming using restarts and sharpness. Math. Program. **201**(1), 133–184 (2023)
4. Basu, K., Ghoting, A., Mazumder, R., Pan, Y.: ECLIPSE: An extreme-scale linear program solver for web-applications. In: International Conference on Machine Learning, pp. 704–714. PMLR (2020)
5. Bauschke, H.H., Combettes, P.L., et al.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces, vol. 408. Springer (2011)
6. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM REV **59**(1), 65–98 (2017)
7. Bredies, K., Chenchene, E., Lorenz, D.A., Naldi, E.: Degenerate preconditioned proximal point algorithms. SIAM J. Optim. **32**(3), 2376–2401 (2022)
8. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB–a quadratic assignment problem library. J. Glob. Optim. **10**, 391–403 (1997)
9. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vision **40**, 120–145 (2011)
10. Chambolle, A., Pock, T.: On the ergodic convergence rates of a first-order primal–dual algorithm. Math. Program. **159**(1), 253–287 (2016)
11. Cui, Y., Li, X., Sun, D.F., Toh, K.C.: On the convergence properties of a majorized alternating direction method of multipliers for linearly constrained convex optimization problems with coupled objective functions. J. Optim. Theory Appl. **169**(3), 1013–1041 (2016)
12. Deng, Q., Feng, Q., Gao, W., Ge, D., Jiang, B., Jiang, Y., Liu, J., Liu, T., Xue, C., Ye, Y., et al.: An enhanced alternating direction method of multipliers-based interior point method for linear and conic optimization. INFORMS J. Comput (2024)
13. Eckstein, J., Bertsekas, D.P.: On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**(1), 293–318 (1992)
14. Esser, E., Zhang, X., Chan, T.F.: A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. SIAM J. Imaging Sci **3**(4), 1015–1046 (2010)
15. Fazel, M., Pong, T.K., Sun, D.F., Tseng, P.: Hankel matrix rank minimization with applications to system identification and realization. SIAM J. Matrix Anal. Appl. **34**(3), 946–977 (2013)
16. Ge, D., Huangfu, Q., Wang, Z., Wu, J., Ye, Y.: Cardinal Optimizer (COPT) User Guide. arXiv preprint arXiv:2208.14314 (2022)

17. Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al.: MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. Math. Program. Comput. **13**(3), 443–490 (2021)
18. Golub, G.H., Van Loan, C.F.: Matrix Computations. JHU press (2013)
19. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2024). URL `https://www.gurobi.com`
20. Halpern, B.: Fixed points of nonexpanding maps. Bull. Am. Math. Soc. **73**(6), 957–961 (1967)
21. Han, D., Sun, D.F., Zhang, L.: Linear rate convergence of the alternating direction method of multipliers for convex composite programming. Math. Oper. Res. **43**(2), 622–637 (2018)
22. IBM: IBM ILOG CPLEX Optimization Studio CPLEX User's Manual (1987)
23. Kim, D.: Accelerated proximal point method for maximally monotone operators. Math. Program. **190**(1), 57–87 (2021)
24. Koch, T., Berthold, T., Pedersen, J., Vanaret, C.: Progress in mathematical programming solvers from 2001 to 2020. EURO J. Comput. Optim. **10**, 100031 (2022)
25. Lieder, F.: On the convergence rate of the Halpern-iteration. Optim. Lett. **15**(2), 405–418 (2021)
26. Lin, T., Ma, S., Ye, Y., Zhang, S.: An ADMM-based interior-point method for large-scale linear programming. Optim. Methods Softw. **36**(2-3), 389–424 (2021)
27. Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**(6), 964–979 (1979)
28. Lu, H., Yang, J.: cuPDLP.jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in julia. arXiv preprint arXiv:2311.12180 (2023)
29. Lu, H., Yang, J.: Restarted Halpern PDHG for linear programming. arXiv preprint arXiv:2407.16144 (2024)
30. Lu, H., Yang, J., Hu, H., Huangfu, Q., Liu, J., Liu, T., Ye, Y., Zhang, C., Ge, D.: cuPDLP-C: A strengthened implementation of cuPDLP for linear programming by C language. arXiv preprint arXiv:2312.14832 (2023)
31. Monteiro, R.D., Svaiter, B.F.: Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. SIAM J. Optim. **23**(1), 475–507 (2013)
32. Nesterov, Y.: Subgradient methods for huge-scale optimization problems. Math. Program. **146**(1), 275–297 (2014)
33. O'Donoghue, B.: Operator splitting for a homogeneous embedding of the linear complementarity problem. SIAM J. Optim. **31**(3), 1999–2023 (2021)
34. O'donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**, 1042–1068 (2016)
35. Pock, T., Chambolle, A.: Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In: 2011 International Conference on Computer Vision, pp. 1762–1769. IEEE (2011)
36. Rockafellar, R.T.: Convex Analysis, vol. 18. Princeton University Press (1970)
37. Ruiz, D.: A scaling algorithm to equilibrate both rows and columns norms in matrices. Tech. rep., Rutherford Appleton Laboratory (2001)
38. Ryu, E.K., Yin, W.: Large-scale Convex Optimization: Algorithms & Analyses via Monotone Operators. Cambridge University Press (2022)
39. Sabach, S., Shtern, S.: A first order method for solving convex bilevel optimization problems. SIAM J. Optim. **27**(2), 640–660 (2017)
40. Shen, L., Pan, S.: Weighted iteration complexity of the sPADMM on the KKT residuals for convex composite optimization. arXiv preprint arXiv:1611.03167 (2016)
41. Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S.: OSQP: An operator splitting solver for quadratic programs. Math. Program. Comput. **12**(4), 637–672 (2020)
42. Sun, D.F., Yuan, Y., Zhang, G., Zhao, X.: Accelerating preconditioned ADMM via degenerate proximal point mappings. arXiv preprint arXiv:2403.18618 (2024). SIAM J. Optim. 35 (2025) XXX, in print.
43. Xu, M., Wu, T.: A class of linearized proximal alternating direction methods. J. Optim. Theory Appl. **151**, 321–337 (2011)

44. Yang, B., Zhao, X., Li, X., Sun, D.F.: An accelerated proximal alternating direction method of multipliers for optimal decentralized control of uncertain systems. J. Optim. Theory Appl. **204**(1), 9 (2025)
45. Zhang, G., Gu, Z., Yuan, Y., Sun, D.F.: HOT: An efficient Halpern accelerating algorithm for optimal transport problems. arXiv preprint arXiv:2408.00598 (2024)
46. Zhang, G., Yuan, Y., Sun, D.F.: An efficient HPR algorithm for the Wasserstein barycenter problem with $O(\mathrm{Dim}(\mathrm{P})/\varepsilon)$ computational complexity. arXiv preprint arXiv:2211.14881 (2022)
47. Zhu, M., Chan, T.: An efficient primal-dual hybrid gradient algorithm for total variation image restoration. UCLA Cam Report **34**(2) (2008)

Statements and Declarations

Funding 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. 15304721. The work of Yancheng Yuan was supported by the Research Center for Intelligent Operations Research and The Hong Kong Polytechnic University under grant P0045485. The work of Xinyuan Zhao was supported in part by the National Natural Science Foundation of China under Project No. 12271015.

Conflict of interest The authors declare that they have no conflict of interest.

Data Availability The datasets analyzed during the current study are available from the corresponding author on reasonable request.