ON THE RELATIONSHIPS AMONG GPU-ACCELERATED FIRST-ORDER METHODS FOR SOLVING LINEAR PROGRAMMING*

KAIHUANG CHEN†, DEFENG SUN‡, YANCHENG YUAN§, GUOJUN ZHANG¶, AND XINYUAN ZHAO∥

Abstract. This paper aims to understand the relationships among recently developed GPU-accelerated first-order methods (FOMs) for linear programming (LP), with particular emphasis on HPR-LP—a Halpern Peaceman–Rachford (HPR) method for LP. Our findings can be summarized as follows: (i) the base algorithm of cuPDLPx, a recently released GPU solver, is a special case of the base algorithm of HPR-LP, thereby showing that cuPDLPx is another concrete implementation instance of HPR-LP; (ii) once the active sets have been identified, HPR-LP and EPR-LP—an ergodic PR method for LP—become equivalent under the same initialization; and (iii) extensive numerical experiments on benchmark datasets demonstrate that HPR-LP achieves the best overall performance among current GPU-accelerated LP solvers. These findings provide a strong motivation for using the HPR method as a baseline to further develop GPU-accelerated LP solvers and beyond.

Key words. GPU acceleration, Linear programming, Halpern Peaceman–Rachford, Proximal alternating direction method of multipliers

MSC codes. 90C05 · 90C06 · 90C25 · 65Y20

1. Introduction. In this paper, we focus on GPU-accelerated first-order methods (FOMs) for solving large-scale linear programming (LP) problems. Our aim is to develop a deep understanding of the connections among recent algorithmic developments and to provide insights that will guide the design of future high-performance LP solvers. Specifically, we consider the following general form of LP:

$$
\begin{aligned}
(1.1) \quad & \min_{x \in \mathbb{R}^n} \quad \langle c, x \rangle \\
& \text{s.t.} \ \ Ax \in \mathcal{K}, \\
& \qquad x \in \mathcal{C},
\end{aligned}
$$

where $c \in \mathbb{R}^n$ is the objective vector, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, $\mathcal{K} := \{s \in \mathbb{R}^m : l_c \leq s \leq u_c\}$ with bounds $l_c \in (\mathbb{R} \cup \{-\infty\})^m$ and $u_c \in (\mathbb{R} \cup \{\infty\})^m$, and $\mathcal{C} := \{x \in \mathbb{R}^n : l_v \leq x \leq u_v\}$ with bounds $l_v \in (\mathbb{R} \cup \{-\infty\})^n$ and $u_v \in (\mathbb{R} \cup \{\infty\})^n$.

†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).

The corresponding dual problem is:

$$
\begin{aligned}
(1.2) \qquad \min_{y \in \mathbb{R}^m,\ z \in \mathbb{R}^n} \quad & \delta_{\mathcal{K}}^*(-y) + \delta_{\mathcal{C}}^*(-z) \\
\text{s.t.} \quad & A^* y + z = c,
\end{aligned}
$$

where $\delta_S^*(\cdot)$ denotes the convex conjugate of the indicator function $\delta_S(\cdot)$ associated with a closed convex set S.

Classical LP algorithms such as the simplex method [13] and interior-point methods [28] relied on factorization-based approaches. Commercial solvers like CPLEX [26] and Gurobi [22] combined simplex and interior-point methods with highly optimized sparse linear algebra, delivering state-of-the-art CPU performance. Yet their reliance on inherently sequential matrix factorizations makes it difficult to fully exploit massively parallel GPUs with high memory bandwidth [37]. Even GPU-accelerated interior-point solvers like CuClarabel [21, 11], which leverage mixed precision, still depend on direct factorizations and thus face scalability challenges on large-scale problems.

In recent years, first-order methods (FOMs) have attracted growing attention for large-scale LPs due to their low per-iteration cost and high parallelizability. Representative solvers include PDLP [1, 3], ECLIPSE [4], ABIP [31, 15], and HPR-LP [8], with related convex conic or quadratic solvers such as SCS [42, 41], OSQP [46], PDQP [38], PDHCG [25], PDCS [32], and HPR-QP [9]. A milestone in this line is PDLP [1, 2], based on the primal dual hybrid gradient (PDHG) method [54, 17, 6], which is equivalent to linearized ADMM with unit stepsize (or linearized Douglas–Rachford) [17, 6]. Its main computational advantage is that it relies only on sparse matrix-vector multiplications and simple projections, avoiding matrix factorizations. Practical enhancements such as restarts with ergodic sequences, adaptive penalty updates, and heuristic line search further improve robustness and efficiency. Notably, PDLP received the 2024 Beale–Orchard-Hays Prize for Excellence in Computational Mathematical Programming. Its GPU implementations, including cuPDLP.jl [35], cuPDLP-C [39], and NVIDIA's cuOpt[1], have demonstrated competitive performance against commercial solvers on large-scale LP benchmarks. Furthermore, instead of relying on costly heuristic line search as in PDLP, Chen et al. [10] established the ergodic convergence of the semi-proximal Peaceman–Rachford (PR) method, which admits larger effective stepsizes at low cost. Building on this foundation, a preliminary implementation of ergodic PR for LP with practical enhancements, termed EPR-LP, has already demonstrated superior empirical performance over cuPDLP.jl [51].

From a theoretical standpoint, ergodic sequences of semi-proximal ADMM, including PDHG, achieve an $O(1/k)$ iteration complexity for objective error and feasibility violation [14, 12]. Recently, building on the ergodic $O(1/k)$ complexity of PDHG established by Chambolle and Pock [6, 7], Applegate et al. [3] proved that PDHG achieves the ergodic $O(1/k)$ complexity for LP problems when measured by primal-dual feasibility violation and the primal-dual gap, both of which can be inferred from the Karush–Kuhn–Tucker (KKT) residual. For the KKT residual itself, Monteiro and Svaiter [40] established an ergodic $O(1/k)$ rate for ADMM (with unit dual stepsize) in terms of the ε-subdifferential. This result was later extended to sPADMM by Shen and Pan [45], and further to the semi-proximal PR method by Chen et al. [10], implying an ergodic $O(1/\sqrt{k})$ complexity for the KKT residual.

---

[1] https://developer.nvidia.com/blog/accelerate-large-linear-programming-problems-with-nvidia-cuopt

In contrast, HPR-LP [8], a Halpern Peaceman–Rachford (HPR) method for LP, achieves a nonergodic $O(1/k)$ rate directly in terms of the KKT residual. It is derived by accelerating the preconditioned (semi-proximal) PR method [53, 50, 47] through the Halpern iteration [23, 30, 44]. Specifically, Zhang et al. [53] applied the Halpern iteration to the classical PR method [16, 33], obtaining the HPR method without proximal terms and establishing an $O(1/k)$ rate for both the KKT residual and the objective error. Subsequently, Sun et al. [47] 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 obtain the HPR method with semi-proximal terms, which also achieves an $O(1/k)$ rate. Building on these advances, Chen et al. [8] developed HPR-LP, a GPU-accelerated solver for large-scale LP that significantly outperformed the award-winning PDLP [1, 3, 35]. Inspired by the work of HPR-LP, Lu and Yang [36] proposed the reflected restarted Halpern PDHG (r^2HPDHG), a special case of the HPR method [47]. Building on this, Lu et al. [34] released an efficient GPU implementation, cuPDLPx, which exhibits superior performance compared to cuPDLP.jl [35].

Despite significant progress in GPU-accelerated FOMs for LP, their interrelationships remain unclear. This paper addresses this gap by investigating the connections among recent approaches, with a focus on HPR-LP and its links to other solvers. The main contributions can be summarized as follows:

1. We prove that the base algorithm of cuPDLPx (r^2HPDHG) is a special case of the base algorithm of HPR-LP, implying that cuPDLPx is another concrete implementation instance of HPR-LP.
2. We show that, under the strict complementarity condition, the PR update in the HPR method for LP becomes affine after finitely many iterations. Consequently, after active sets are identified, HPR-LP and EPR-LP are equivalent under consistent initialization based on the equivalence between Halpern iteration and ergodic iteration for affine fixed-point maps.
3. We present extensive experiments on LP benchmark datasets, evaluating the impact of algorithmic enhancements and demonstrating that HPR-LP achieves the best overall performance among current GPU-accelerated LP solvers.

The remainder of this paper is organized as follows. Section 2 briefly reviews the HPR method for solving LP. Section 3 discusses the relationship between cuPDLPx and HPR-LP as well as the connection between HPR-LP and EPR-LP. Section 4 presents extensive numerical results on recent GPU-accelerated FOMs for LP across different benchmark sets. Finally, Section 5 concludes the paper.

Notation. Let $\mathbb{R}^n$ be the n-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$. For $A \in \mathbb{R}^{m \times n}$, let A^* be its transpose and $\|A\| := \sqrt{\lambda_1(AA^*)}$ its spectral norm, where $\lambda_1(\cdot)$ denotes the largest eigenvalue. For any self-adjoint positive semidefinite operator $\mathcal{M} : \mathbb{R}^n \to \mathbb{R}^n$, define the seminorm $\|x\|_{\mathcal{M}} := \sqrt{\langle x, \mathcal{M}x \rangle}$. For a convex function $f : \mathbb{R}^n \to (-\infty, +\infty]$, let $\partial f(\cdot)$ be its subdifferential and $\text{Prox}_f(x) := \arg\min_{z \in \mathbb{R}^n} \left\{ f(z) + \frac{1}{2}\|z - x\|^2 \right\}$ its proximal mapping. For a convex set $C \subseteq \mathbb{R}^n$, we use the following notations: the indicator function $\delta_C(x) := 0$ if $x \in C$ and $+\infty$ otherwise; the distance under $\| \cdot \|_{\mathcal{M}}$, $\text{dist}_{\mathcal{M}}(x, C) := \inf_{z \in C} \|x - z\|_{\mathcal{M}}$; the Euclidean projection $\Pi_C(x) := \arg\min_{z \in C} \|x - z\|$; and the normal cone at $x \in C$, denoted by $\mathcal{N}_C(x)$.

2. HPR-LP: An HPR method for solving LP. In this section, we begin by presenting the base algorithm of HPR-LP [8], followed by a discussion of its con-

vergence guarantees and complexity results, which motivate subsequent algorithmic enhancements.

2.1. Base algorithm. For any $(y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$, the augmented Lagrangian of the dual problem (1.2) is

$$L_\sigma(y, z; x) := \delta_\mathcal{K}^*(-y) + \delta_\mathcal{C}^*(-z) + \langle x, A^*y + z - c \rangle + \frac{\sigma}{2}\|A^*y + z - c\|^2,$$

where $\sigma > 0$ is a penalty parameter. For notational convenience, let $w := (y, z, x) \in \mathbb{W} := \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$. Then, an HPR method with semi-proximal terms [47, 8] for solving problems (1.1) and (1.2) is summarized in Algorithm 2.1.

Algorithm 2.1 An HPR method with semi-proximal terms for the problem (1.2)

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 $\mathcal{T}_1 + AA^*$ is positive definite. Denote $w = (y, z, x)$ and $\bar{w} = (\bar{y}, \bar{z}, \bar{x})$. Choose an initial point $w^0 = (y^0, z^0, x^0) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$.

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

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 $\{\bar{w}^k\}$.

Remark 2.1. Steps 1–3 correspond to the semi-proximal DR method [20, 18]. Adding Step 4 (relaxation) yields the semi-proximal PR method, and Step 5 introduces Halpern iteration with stepsize $1/(k + 2)$ [23, 30]. Together, Algorithm 2.1 is an accelerated preconditioned ADMM (pADMM) with parameters $\alpha = 2$ and $\rho = 2$ [47].

According to [43, Corollary 28.3.1], a pair $(y^*, z^*) \in \mathbb{R}^m \times \mathbb{R}^n$ is an optimal solution to problem (1.2) if there exists $x^* \in \mathbb{R}^n$ such that (y^*, z^*, x^*) satisfies the following KKT system:

$$(2.1) \qquad 0 \in Ax^* - \partial\delta_\mathcal{K}^*(-y^*), \quad 0 \in x^* - \partial\delta_\mathcal{C}^*(-z^*), \quad A^*y^* + z^* - c = 0.$$

We make the following assumption:

ASSUMPTION 2.2. *There exists a vector* $(y^*, z^*, x^*) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ *satisfying the KKT system* (2.1).

Under Assumption 2.2, solving the primal–dual pair (1.1)–(1.2) is equivalent to finding a point $w^* \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ such that $0 \in \mathcal{T}w^*$, where the maximal monotone operator $\mathcal{T}$ is defined as

$$(2.2) \qquad \mathcal{T}w = \begin{pmatrix} -\partial\delta_\mathcal{K}^*(-y) + Ax \\ -\partial\delta_\mathcal{C}^*(-z) + x \\ c - A^*y - z \end{pmatrix} \quad \forall w = (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n.$$

The global convergence of Algorithm 2.1 is established in the following proposition.

PROPOSITION 2.3 (Corollary 3.5 in [47]). *Suppose that Assumption 2.2 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 2.1 converges to a point $w^* = (y^*, z^*, x^*)$, where (y^*, z^*) solves the dual problem (1.2) and x^* solves the primal problem (1.1).*

Next, consider the self-adjoint positive semidefinite linear operator $\mathcal{M} : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ defined by

$$(2.3) \qquad \mathcal{M} = \begin{bmatrix} \sigma AA^* + \sigma\mathcal{T}_1 & 0 & A \\ 0 & 0 & 0 \\ A^* & 0 & \frac{1}{\sigma}I_n \end{bmatrix},$$

where I_n denotes the $n \times n$ identity matrix. To analyze the complexity of the HPR method with semi-proximal terms, we consider the KKT residual and the objective error. The residual mapping associated with the KKT system (2.1), as introduced in [24], is given by

$$(2.4) \qquad \mathcal{R}(w) = \begin{pmatrix} Ax - \Pi_{\mathcal{K}}(Ax - y) \\ x - \Pi_{\mathcal{C}}(x - z) \\ c - A^*y - z \end{pmatrix} \qquad \forall w = (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n.$$

Furthermore, let $\{(\bar{y}^k, \bar{z}^k)\}$ be the sequence generated by Algorithm 2.1. We define the objective error as

$$h(\bar{y}^{k+1}, \bar{z}^{k+1}) := \delta_{\mathcal{K}}^*(-\bar{y}^{k+1}) + \delta_{\mathcal{C}}^*(-\bar{z}^{k+1}) - \delta_{\mathcal{K}}^*(-y^*) - \delta_{\mathcal{C}}^*(-z^*) \quad \forall k \geq 0,$$

where (y^*, z^*) is the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k)\}$. The complexity of the HPR method with semi-proximal terms is summarized in Theorem 2.4.

THEOREM 2.4 (Proposition 2.9 and Theorem 3.7 in [47]). *Suppose that Assumption 2.2 holds. Let $\{w^k\} = \{(y^k, z^k, x^k)\}$ and $\{\bar{w}^k\} = \{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ be two sequences generated by the HPR method with semi-proximal terms in Algorithm 2.1, and let $w^* = (y^*, z^*, x^*)$ be its limit point. Define $R_0 = \|w^0 - w^*\|_{\mathcal{M}}$. Then for all $k \geq 0$, the following iteration complexity bounds hold:*

$$(2.5a) \qquad \|\bar{w}^{k+1} - w^k\|_{\mathcal{M}} \leq \frac{R_0}{k+1},$$

$$(2.5b) \qquad \|\mathcal{R}(\bar{w}^{k+1})\| \leq \left(\frac{\sigma(\|A\| + \|\sqrt{\mathcal{T}_1}\|) + 1}{\sqrt{\sigma}} \right) \frac{R_0}{k+1},$$

$$(2.5c) \qquad \left(-\frac{1}{\sqrt{\sigma}}\|x^*\| \right) \frac{R_0}{k+1} \leq h(\bar{y}^{k+1}, \bar{z}^{k+1}) \leq \left(3R_0 + \frac{1}{\sqrt{\sigma}}\|x^*\| \right) \frac{R_0}{k+1}.$$

The results above establish that the HPR method (Algorithm 2.1) enjoys an $O(1/k)$ complexity rate in terms of the KKT residual and the objective error. These properties motivate the use of restart strategies and adaptive parameter updates, which will be discussed in the next subsection.

2.2. Algorithmic enhancements. Several enhancements have been proposed to improve the performance of the HPR method for solving LP [8] and convex composite quadratic programming (CCQP) [9]. In particular, restart strategies and adaptive updates of the penalty parameter σ, motivated by the $O(1/k)$ complexity results in Theorem 2.4, have proven effective. For completeness, we summarize the HPR-LP framework with these enhancements in Algorithm 2.2.

Algorithm 2.2 HPR-LP: A Halpern Peaceman–Rachford method for the LP problem (1.2) (cf. [8])

1: **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 $w^{0,0} = (y^{0,0}, z^{0,0}, x^{0,0}) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$.

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

3: **repeat**

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

5:    **repeat**

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

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

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

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

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

11:     $t = t + 1, \ k = k + 1$;

12:    **until** one of the restart criteria holds or all termination criteria hold

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

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

15: **until** termination criteria hold

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

2.2.1. Restart strategy. Restarting has been recognized as particularly important for Halpern iterations. As noted in Theorem 2.4, the complexity bound depends on the weighted distance R_0 between the initial point and the optimal solution. Consequently, as the iterates approach optimality, continuing to reference a distant initial anchor becomes counterproductive, whereas resetting the anchor to the current iterate helps reduce the bound and refocus the iteration near the solution. This observation motivates the merit function

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

where w^* is any solution of the KKT system (2.1). Since w^* is unknown, the practical surrogate

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

is employed in defining restart rules. The following criteria are commonly adopted:

   1. **Sufficient decay:**

(2.6) $$\widetilde{R}_{r,t+1} \leq \alpha_1 \widetilde{R}_{r,0};$$

   2. **Necessary decay + no local progress:**

(2.7) $$\widetilde{R}_{r,t+1} \leq \alpha_2 \widetilde{R}_{r,0}, \quad \text{and} \quad \widetilde{R}_{r,t+1} > \widetilde{R}_{r,t};$$

3. **Long inner loop:**

$$(2.8) \qquad\qquad\qquad\qquad t \geq \alpha_3 k;$$

where $0 < \alpha_1 < \alpha_2 < 1$ and $0 < \alpha_3 < 1$. When any criterion is met, the inner loop is restarted at iteration $(r+1)$ with $w^{r+1,0} = \bar{w}^{r,\tau_r}$ and an updated σ_{r+1}.

Remark 2.5. Restart strategies are commonly used in first-order methods for LP [1, 35, 39, 36]. For instance, PDLP adopts a normalized duality gap as the merit function [1], while subsequent works by Lu et al. [35, 39] introduced variants based on weighted KKT residuals.

2.2.2. Update rules for σ. Another important enhancement of HPR methods concerns the update of the penalty parameter σ. The update strategy is motivated by the complexity results of the HPR method in Algorithm 2.1 (see Theorem 2.4). At a high level, the goal is to select σ at each restart to tighten the complexity bound and thereby reduce the KKT residuals in subsequent iterations. Specifically, the ideal update is defined as the minimizer of the weighted distance to the optimal solution:

$$(2.9) \qquad\qquad\qquad \sigma_{r+1} := \arg\min_{\sigma} \left\| w^{r+1,0} - w^* \right\|_{\mathcal{M}}^2 ,$$

where w^* is any solution of the KKT system (2.1). Substituting the definition of $\mathcal{M}$ from (2.3) leads to the closed-form expression

$$(2.10) \qquad\qquad \sigma_{r+1} = \sqrt{\frac{\|x^{r+1,0} - x^*\|^2}{\|y^{r+1,0} - y^*\|_{\mathcal{T}_1}^2 + \|A^*(y^{r+1,0} - y^*)\|^2}}.$$

Since the optimal solution (x^*, y^*) is unknown, practical implementations approximate these terms using the observed progress within each outer loop:

$$(2.11) \qquad \Delta_x := \|\bar{x}^{r,\tau_r} - x^{r,0}\|, \qquad \Delta_y := \sqrt{\|\bar{y}^{r,\tau_r} - y^{r,0}\|_{\mathcal{T}_1}^2 + \|A^*(\bar{y}^{r,\tau_r} - y^{r,0})\|^2},$$

which yields the implementable update rule

$$(2.12) \qquad\qquad\qquad\qquad \sigma_{r+1} = \frac{\Delta_x}{\Delta_y}.$$

Several special cases of $\mathcal{T}_1$ have been investigated in the literature [8]:

1. **Case $\mathcal{T}_1 = 0$.** This case occurs when $b := l_c = u_c$, which arises in applications with special structure in A, such as optimal transport [52] and Wasserstein barycenter problem [53]. The y-update then reduces to solving the linear system

$$(2.13) \qquad AA^* \bar{y}^{r,t+1} = \frac{1}{\sigma_r}\big(b - A(\bar{x}^{r,t+1} + \sigma_r(\bar{z}^{r,t+1} - c))\big),$$

which is computationally affordable in practice. In this case, the update rule (2.12) simplifies to

$$(2.14) \qquad\qquad\qquad \sigma_{r+1} = \frac{\|\bar{x}^{r,\tau_r} - x^{r,0}\|}{\|A^*(\bar{y}^{r,\tau_r} - y^{r,0})\|}.$$

7

2. **Case $\mathcal{T}_1 = \lambda_A I_m - AA^*$ with $\lambda_A \geq \|A\|^2$.** Proposed in [17, 6, 49], this choice applies when $l_c \neq u_c$ or when solving (2.13) directly is expensive. The y-update takes the form

$$(2.15) \qquad \bar{y}^{r,t+1} = \frac{1}{\sigma_r \lambda_A}\Big(\Pi_\mathcal{K}(R_y) - R_y\Big),$$

where $R_y := A(2\bar{x}^{r,t+1} - x^{r,t}) - \sigma_r \lambda_A y^{r,t}$. In this setting, the update for σ becomes

$$(2.16) \qquad \sigma_{r+1} = \frac{1}{\sqrt{\lambda_A}}\frac{\|\bar{x}^{r,\tau_r} - x^{r,0}\|}{\|\bar{y}^{r,\tau_r} - y^{r,0}\|}.$$

Remark 2.6. The update formula (2.16) is closely related to the primal weight update in PDLP [1, Algorithm 3], differing mainly by the presence of the factor λ_A.

It is worth noting that the approximations Δ_x and Δ_y may deviate significantly from the true quantities. To address this, various smoothing schemes [1, 35, 39, 9] and safeguards [8] have been proposed to stabilize the update. Moreover, the rule (2.12) is not specific to LP; it extends directly to the HPR method for more general convex optimization problems [47], including CCQP [9].

3. Relationships among GPU-accelerated LP solvers. Fig. 3.1 summarizes the connections among several popular FOMs for solving LP. Two relationships are of particular interest in this paper. First, we demonstrate that the base algorithm of cuPDLPx (r^2HPDHG) [34] is a special case of HPR-LP under suitable parameter choices. Second, we examine the relationship between HPR-LP and EPR-LP, focusing on the interplay between the Halpern iteration and the ergodic iteration. The following subsections elaborate on these two relationships in detail.

stepsize=1
$T_1 = \lambda_1(AA^*)I - AA^*$

PDHG

(Semi-proximal)
pADMM

Equivalent
for affine fixed-point maps

Ergodic iterations

Halpern iterations
(stepsize 1/(k+2))

$T_1 = \lambda_1(AA^*)I - AA^*$

Ergodic PDHG

Ergodic DR
(EDR)

Halpern DR
(HDR)

Algorithmic
enhancements

Precondition

Restart

Primal weight update

Line search

Relaxation

EPR

Algorithmic
enhancements

Precondition

Restart

Sigma update

HPR

PDLP

ERR-LP

HRR-LP

cuPDLP.jl

cuPDLP-C

cuOpt

cuPDLPx

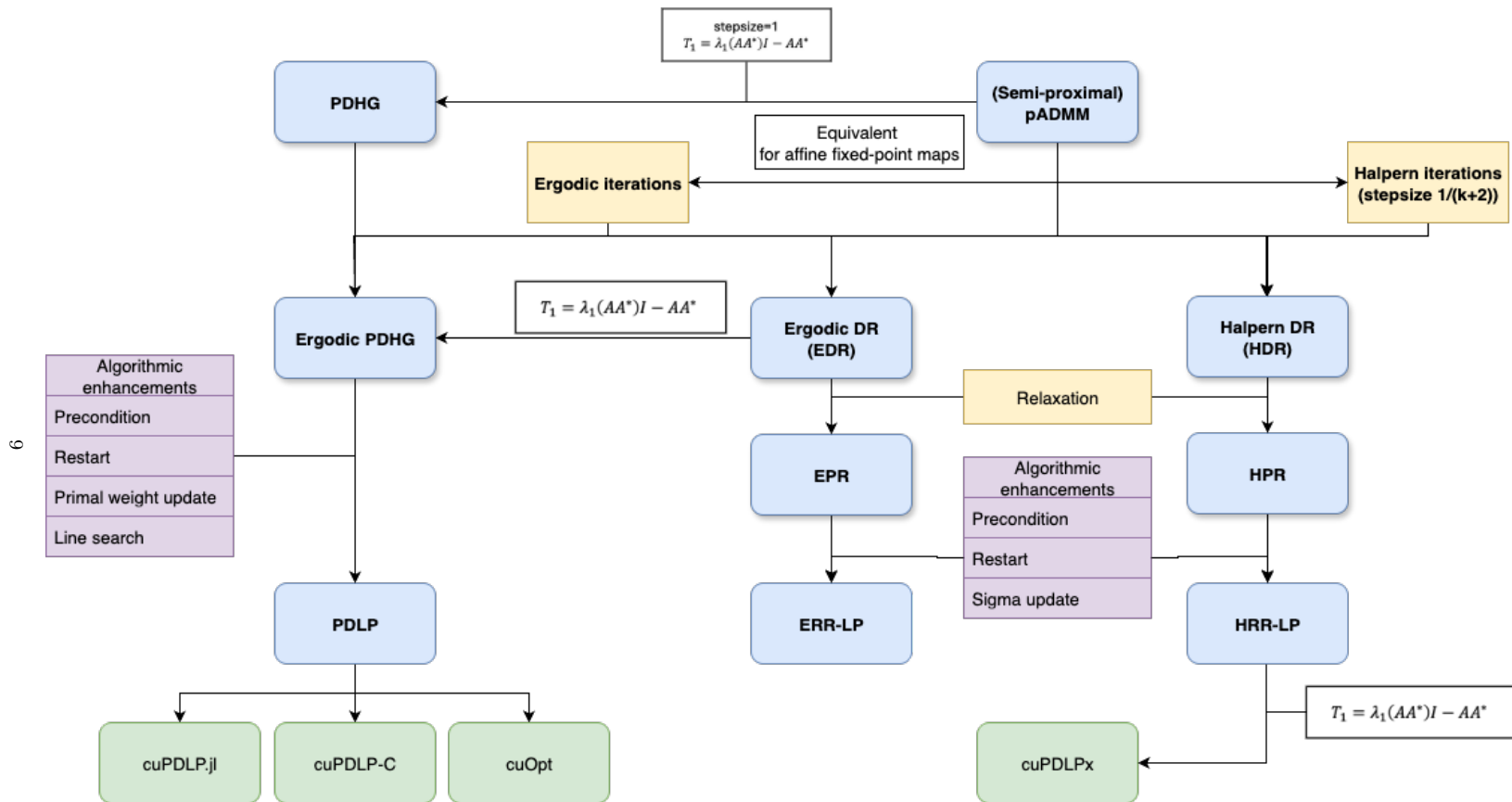$T_1 = \lambda_1(AA^*)I - AA^*$

Fig. 3.1: Relationships among GPU-accelerated FOMs for solving LP.

3.1. The relationship between cuPDLPx and HPR-LP. Lu et al. [34] recently released cuPDLPx, a GPU-accelerated solver that enhances PDLP [1] by incorporating the Halpern iteration [23, 30]. The key observation is that the base iteration of cuPDLPx (r^2HPDHG) is a special case of the HPR method (Algorithm 2.1). Specifically, starting from an initial point $u^0 = (y^0, x^0) \in \mathbb{R}^m \times \mathbb{R}^n$, the base iteration of cuPDLPx with reflection parameter $\gamma \in [0, 1]$ is given by

$$
(3.1) \quad
\begin{cases}
\mathrm{PDHG}(u^k) =
\begin{cases}
\bar{x}^{k+1} = \Pi_{\mathcal{C}}\left(x^k - \frac{\eta}{\omega}(c - A^* y^k)\right), \\
\bar{y}^{k+1} = y^k - \eta\omega A(2\bar{x}^{k+1} - x^k) \\
\qquad\quad -\eta\omega\Pi_{-\mathcal{K}}\left(\frac{1}{\eta\omega}y^k - A(2\bar{x}^{k+1} - x^k)\right),
\end{cases} \\
u^{k+1} = \frac{k+1}{k+2}\left((1+\gamma)\,\mathrm{PDHG}(u^k) - \gamma u^k\right) + \frac{1}{k+2}u^0,
\end{cases}
$$

where η is the stepsize and ω is the primal weight. The relationship between cuPDLPx and the HPR method is stated below.

PROPOSITION 3.1. *The sequence $\{(x^k, y^k)\}$ generated by cuPDLPx in (3.1) with $\gamma = 1$ coincides with the sequence produced by the HPR method (Algorithm 2.1) with $\mathcal{T}_1 = \lambda_A I_m - AA^*$, provided that*

$$
\sigma = \frac{\eta}{\omega}, \qquad \lambda_A = \frac{1}{\eta^2} \geq \|A\|^2,
$$

and both algorithms are initialized at the same point.

Proof. Taking $\mathcal{T}_1 = \lambda_A I_m - AA^*$ with $\lambda_A \geq \|A\|_2^2$, Algorithm 2.1 simplifies to:

$$
(3.2) \quad
\begin{cases}
\bar{z}^{k+1} = \frac{1}{\sigma}\left(\Pi_{\mathcal{C}}\left(x^k + \sigma(A^* y^k - c)\right) - \left(x^k + \sigma(A^* y^k - c)\right)\right), \\
\bar{x}^{k+1} = x^k + \sigma(A^* y^k + \bar{z}^{k+1} - c) = \Pi_{\mathcal{C}}\left(x^k + \sigma(A^* y^k - c)\right), \\
\bar{y}^{k+1} = \frac{1}{\sigma\lambda_A}\left(\Pi_{\mathcal{K}}\left(A(2\bar{x}^{k+1} - x^k) - \sigma\lambda_A y^k\right) - \left(A(2\bar{x}^{k+1} - x^k) - \sigma\lambda_A y^k\right)\right), \\
\hat{w}^{k+1} = 2\bar{w}^{k+1} - w^k, \\
w^{k+1} = \frac{1}{k+2}w^0 + \frac{k+1}{k+2}\hat{w}^{k+1}.
\end{cases}
$$

Comparing the update schemes (3.1) with $\gamma = 1$ and (3.2), we observe that the two methods produce identical iterates when the parameters satisfy the relations

$$
\frac{\eta}{\omega} = \sigma, \qquad \eta\omega = \frac{1}{\lambda_A \sigma},
$$

and both are initialized from the same point. That is, if

$$
(3.3) \quad \sigma = \frac{\eta}{\omega}, \qquad \lambda_A = \frac{1}{\eta^2} \geq \|A\|^2,
$$

then the sequence $\{u^k = (y^k, x^k)\}$ generated by (3.1) with $\gamma = 1$ coincides with the sequence $\{(y^k, x^k)\}$ produced by (3.2). $\qquad\square$

Remark 3.2. In [34], the stepsize is chosen as $\eta = 0.998/\|A\|$. By the above equivalence, one may instead take $\eta = 1/\|A\|$ to obtain a larger stepsize, thereby improving performance while preserving theoretical guarantees established in [47]. Furthermore, for $\gamma \in [0,1)$, the scheme (3.1) can be viewed as a special case of the Halpern-accelerated pADMM with parameter $\rho = \gamma + 1$ [47].

3.2. The relationship between HPR-LP and EPR-LP. In this subsection, we establish the connection between the base algorithms of HPR-LP and EPR-LP through the relationship between ergodic and Halpern iterations. To this end, we first recall the EPR method proposed in [10] for solving the dual problem (1.2), summarized in Algorithm 3.1.

Algorithm 3.1 An EPR method for the LP problem (1.2) [10]

1: **Input:** Set $\sigma > 0$ and choose a self-adjoint positive semidefinite operator $\mathcal{T}_1 : \mathbb{R}^m \to \mathbb{R}^m$ such that $\mathcal{T}_1 + AA^*$ is positive definite. Let $w = (y, z, x)$, $\bar{w} = (\bar{y}, \bar{z}, \bar{x})$, and initialize $w^0 = (y^0, z^0, x^0) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Step 1: $\bar{z}^{k+1} = \arg\min\limits_{z \in \mathbb{R}^n} L_\sigma(y^k, z; x^k)$;
4: Step 2: $\bar{x}^{k+1} = x^k + \sigma(A^* y^k + \bar{z}^{k+1} - c)$;
5: Step 3: $\bar{y}^{k+1} = \arg\min\limits_{y \in \mathbb{R}^m} \{L_\sigma(y, \bar{z}^{k+1}; \bar{x}^{k+1}) + \frac{\sigma}{2}\|y - y^k\|^2_{\mathcal{T}_1}\}$;
6: Step 4: $w^{k+1} = 2\bar{w}^{k+1} - w^k$;
7: **end for**
8: **Output:** $\{\bar{w}_a^{k+1} = \frac{1}{k+2}\sum_{j=0}^{k+1} \bar{w}^{j+1}\}$ and $\{w_a^{k+1} = \frac{1}{k+2}\sum_{j=0}^{k+1} w^{j+1}\}$.

Remark 3.3. Chen et al. [10] proved that the ergodic sequence $\{\bar{w}_a^{k+1}\}$ generated by the EPR method converges for convex optimization problems with linear constraints. They also provided an example showing that the ergodic sequence $\{w_a^{k+1}\}$ fails to converge while the sequence $\{w^k\}$ is unbounded.

We now formalize the identity that the Halpern iteration with stepsize $1/(k+2)$ coincides with uniform ergodic averages of Picard iterates when the underlying mapping is affine. This observation goes back to the analysis of Halpern iteration [48] and has been made explicit in, e.g., [29].

PROPOSITION 3.4. *Let $\mathcal{F} : \mathbb{W} \to \mathbb{W}$ be an affine map of the form*

$$\mathcal{F}(w) = Rw + b,$$

where $R : \mathbb{W} \to \mathbb{W}$ is linear and $b \in \mathbb{W}$ is fixed. Given $w^0 \in \mathbb{W}$, consider the Halpern iteration

$$w^{k+1} = \frac{1}{k+2} w^0 + \frac{k+1}{k+2} \mathcal{F}(w^k) \qquad \forall k \geq 0.$$

Then, for every $k \geq 0$,

$$w^{k+1} = \frac{1}{k+2} \sum_{j=0}^{k+1} \mathcal{F}^j(w^0),$$

i.e., w^{k+1} equals the ergodic average of the affine Picard iterates $\{\mathcal{F}^j(w^0)\}_{j=0}^{k+1}$.

Proof. Let $p_j := \mathcal{F}^j(w^0)$ for $j \geq 0$. We prove by induction on k that

(3.4)
$$w^{k+1} = \frac{1}{k+2} \sum_{j=0}^{k+1} p_j.$$

For case $k = 0$, we have $w^1 = \frac{1}{2} w^0 + \frac{1}{2} \mathcal{F}(w^0) = \frac{1}{2}(p_0 + p_1)$, which is (3.4). Assume (3.4) holds for some $k \geq 0$, i.e.,

$$w^k = \frac{1}{k+1} \sum_{j=0}^{k} p_j.$$

Using the Halpern update and the affinity of $\mathcal{F}$ on convex combinations,

$$w^{k+1} = \frac{1}{k+2} w^0 + \frac{k+1}{k+2} \mathcal{F}(w^k) = \frac{1}{k+2} p_0 + \frac{k+1}{k+2} \mathcal{F}\Big(\frac{1}{k+1} \sum_{j=0}^{k} p_j\Big)$$

$$= \frac{1}{k+2} p_0 + \frac{k+1}{k+2} \Big(\frac{1}{k+1} \sum_{j=0}^{k} \mathcal{F}(p_j)\Big) = \frac{1}{k+2} \Big(p_0 + \sum_{j=0}^{k} p_{j+1}\Big) = \frac{1}{k+2} \sum_{j=0}^{k+1} p_j,$$

which is (3.4). Hence, the identity holds for all $k \geq 0$. $\qquad\square$

We next state a sufficient condition under which the PR updates for LP in Algorithm 2.1 reduce to an affine fixed-point map. We adopt the following strict complementarity notion for a KKT point (x^*, y^*, z^*): for each i,

(3.5)
$$x_i^* \in \{(l_v)_i, (u_v)_i\} \;\Rightarrow\; |z_i^*| > 0, \qquad x_i^* \in ((l_v)_i, (u_v)_i) \;\Rightarrow\; z_i^* = 0,$$

and for each j,

(3.6)
$$(Ax^*)_j \in \{(l_c)_j, (u_c)_j\} \;\Rightarrow\; |y_j^*| > 0, \qquad (Ax^*)_j \in ((l_c)_j, (u_c)_j) \;\Rightarrow\; y_j^* = 0.$$

The associated optimal active index sets are defined as

$$I_C^* := \{\, i: \; x_i^* \in \{(l_v)_i, (u_v)_i\} \,\}, \qquad I_K^* := \{\, j: \; (Ax^*)_j \in \{(l_c)_j, (u_c)_j\} \,\}.$$

In the $\bar{x}$–update of Algorithm 2.1, we set for any $k \geq 0$,

(3.7)
$$\xi^k := x^k + \sigma(A^* y^k - c), \qquad \bar{x}^{k+1} = \Pi_{\mathcal{C}}(\xi^k),$$

and define the projection–active indices

$$I_C^k := \{\, i: \; [\Pi_{\mathcal{C}}(\xi^k)]_i \in \{(l_v)_i, (u_v)_i\} \,\} \quad \forall k \geq 0.$$

Similarly, in the $\bar{y}$–update of Algorithm 2.1 with $\mathcal{T}_1 = \lambda_A I_m - AA^*$ and $\lambda_A \geq \|A\|^2$, for any $k \geq 0$, we let

(3.8)
$$\zeta^k := A(2\bar{x}^{k+1} - x^k) - \sigma \lambda_A y^k, \qquad \bar{y}^{k+1} = \frac{1}{\sigma \lambda_A}\big(\Pi_{\mathcal{K}}(\zeta^k) - \zeta^k\big),$$

and define

$$I_K^k := \{\, j: \; [\Pi_{\mathcal{K}}(\zeta^k)]_j \in \{(l_c)_j, (u_c)_j\} \,\} \quad \forall k \geq 0.$$

We now show that under the strict complementarity condition, the active sets can be identified in finitely many steps, after which the PR map reduces to an affine operator on the corresponding face.

PROPOSITION 3.5 (Finite identification of projection-active sets). *Suppose Assumption 2.2 holds. Let $\mathcal{T}_1 = \lambda_A I_m - AA^*$ with $\lambda_A \geq \|A\|^2$. Assume the sequence $\{(\bar{x}^k, \bar{y}^k, \bar{z}^k)\}$ generated by Algorithm 2.1 converges to a solution (x^*, y^*, z^*) satisfying strict complementarity conditions in (3.5) and (3.6). Then there exists $K < \infty$ such that for all $k \geq K$,*

$$I_C^k = I_C^*, \qquad I_K^k = I_K^*.$$

Proof. Step 1 (margins). Strict complementarity ensures separation from degeneracy, which implies

$$\gamma_C := \min_{i \in I_C^*} |z_i^*|, \quad \gamma_K := \min_{j \in I_K^*} |y_j^*|, \quad \gamma := \min\{\gamma_C, \gamma_K\} > 0.$$

For interior components, define

$$\begin{cases} \beta_C := \min_{i \notin I_C^*} \min\{ x_i^* - (l_v)_i, \ (u_v)_i - x_i^* \} > 0, \\ \beta_K := \min_{j \notin I_K^*} \min\{ (Ax^*)_j - (l_c)_j, \ (u_c)_j - (Ax^*)_j \} > 0. \end{cases}$$

Step 2 (identification for Π_C). Using $A^*y^* + z^* = c$, set

$$\xi^* := x^* + \sigma(A^*y^* - c) = x^* - \sigma z^*.$$

From (3.7) and Algorithm 2.1, we have $\xi^k = \bar{x}^{k+1} - \sigma\bar{z}^{k+1} \to \xi^*$. If $i \in I_C^*$, then: - if $x_i^* = (l_v)_i$ with $z_i^* \geq \gamma$, we have $\xi_i^* \leq (l_v)_i - \sigma\gamma$, hence $[\Pi_C(\xi^*)]_i = (l_v)_i$; - if $x_i^* = (u_v)_i$ with $z_i^* \leq -\gamma$, we have $\xi_i^* \geq (u_v)_i + \sigma\gamma$, hence $[\Pi_C(\xi^*)]_i = (u_v)_i$. By continuity of Π_C, there exists $K_{1,a} < \infty$ such that for all $k \geq K_{1,a}$, $[\Pi_C(\xi^k)]_i$ remains at the corresponding bound, i.e., $i \in I_C^k$. If $i \notin I_C^*$, then x_i^* lies at least β_C away from the bounds, so $[\Pi_C(\xi^*)]_i = x_i^*$ is strictly interior. Since $\xi^k \to \xi^*$, there exists $K_{1,b} < \infty$ such that for all $k \geq K_{1,b}$, $[\Pi_C(\xi^k)]_i$ also stays strictly interior, i.e., $i \notin I_C^k$. Taking $K_1 = \max\{K_{1,a}, K_{1,b}\}$ gives $I_C^k = I_C^*$ for all $k \geq K_1$.

Step 3 (identification for Π_K). Define $\zeta^* := Ax^* - \sigma\lambda_A y^*$. From (3.8) and Algorithm 2.1, we have

$$\zeta^k = A\bar{x}^{k+1} + \sigma A\bar{z}^{k+1} + \sigma(AA^* - \lambda_A I_m)y^k - \sigma Ac \quad \forall k \geq 0.$$

Since $(\bar{x}^{k+1}, \bar{z}^{k+1}) \to (x^*, z^*)$ and $\bar{y}^k \to y^*$, and Theorem 2.4 ensures that $\|\bar{y}^{k+1} - y^k\|_{\lambda_A I_m - AA^*} \to 0$, it follows that $(AA^* - \lambda_A I_m)y^k \to (AA^* - \lambda_A I_m)y^*$ and hence $\zeta^k \to \zeta^*$. If $j \in I_K^*$, then $(Ax^*)_j = (l_c)_j$ with $y_j^* \geq \gamma$ implies $\zeta_j^* \leq (l_c)_j - \sigma\lambda_A\gamma$ and thus $[\Pi_K(\zeta^*)]_j = (l_c)_j$; the upper-bound case is analogous. By continuity of Π_K, there exists $K_{2,a} < \infty$ such that $j \in I_K^k$ for all $k \geq K_{2,a}$. If $j \notin I_K^*$, then $(Ax^*)_j$ lies at least β_K away from the bounds, so $[\Pi_K(\zeta^*)]_j = (Ax^*)_j$ is interior; since $\zeta^k \to \zeta^*$, there exists $K_{2,b} < \infty$ such that $j \notin I_K^k$ for all $k \geq K_{2,b}$. Taking $K_2 = \max\{K_{2,a}, K_{2,b}\}$ yields $I_K^k = I_K^*$ for all $k \geq K_2$.

Step 4 (conclusion). Taking $K = \max\{K_1, K_2\}$ completes the proof. □

The next corollary shows that after active sets have been identified, the base algorithms of HPR-LP (Algorithm 2.1) and EPR-LP (Algorithm 3.1) coincide.

COROLLARY 3.6 (HPR–EPR equivalence under fixed active sets). *Suppose Assumption 2.2 holds and let $\mathcal{T}_1 = \lambda_A I_m - AA^*$ with $\lambda_A \geq \|A\|^2$. Assume that $I_C^k = I_C^*$ and $I_K^k = I_K^*$ for all $k \geq 0$. Suppose the HPR method (Algorithm 2.1) and the EPR method (Algorithm 3.1) use the same parameters (σ, λ_A) and are initialized at the*

same point w^0. Then the sequence $\{w^{k+1}\}$ generated by the HPR method coincides with the ergodic sequence $\{w_a^{k+1}\}$ generated by the EPR method.

Proof. Since $I_C^k = I_C^*$ and $I_K^k = I_K^*$ for all $k \geq 0$, the projections $\Pi_\mathcal{C}$ and $\Pi_\mathcal{K}$ act on fixed faces. Each projection is affine on these faces, so the one-step PR map $w^k \mapsto \hat{w}^{k+1}$ in Algorithm 2.1 is affine. By Proposition 3.4, the Halpern iteration (HPR) coincides with the ergodic average of the Picard iterates (EPR), hence the sequences coincide. □

Remark 3.7. Proposition 3.5 shows that under the strict complementarity condition, there exists $K < \infty$ such that $I_C^k = I_C^*$ and $I_K^k = I_K^*$ for all $k \geq K$. If both the HPR method (Algorithm 2.1) and the EPR method (Algorithm 3.1) are initialized at $\bar{w}^K$ and this initialization does not change the active sets (I_C^*, I_K^*), then from that point onward the iterates evolve on the fixed affine face determined by these sets. Thus, the assumption on active sets in Corollary 3.6 can be realized after finitely many iterations by shifting the time index, which justifies the equivalence result in practice.

Remark 3.8. A related observation was made by Lu et al. [36], who proved that in unconstrained bilinear problems the ergodic sequence of PDHG is identical to that of its Halpern-accelerated variant.

4. Numerical experiments. In this section, we present extensive numerical results on standard LP benchmark datasets. We begin with evaluating the impact of individual algorithmic components, followed by comparisons with state-of-the-art GPU-accelerated solvers.

4.1. Experimental setup.
Computing environment. 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 RAM.

Datasets. We evaluate the solvers on two standard LP benchmark datasets. (i) The Mittelmann LP benchmark[2], which is widely used for testing both commercial and open-source LP solvers. (ii) A subset of 18 large-scale instances from MIPLIB 2017 [19], where the number of nonzeros in A exceeds 10^7, and we solve their LP relaxations.

Shifted geometric mean. To evaluate solver performance across instances, we report the shifted geometric mean (SGM) with shift $\Delta = 10$, as in the Mittelmann 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 in seconds for the i-th instance. Unsolved instances are assigned a time limit.

Termination criteria. In HPR-LP, the sequence $\{\bar{w}^{r,t}\}$ is used to check the stopping criteria. We terminate HPR-LP when the following stopping criteria are satisfied for the tolerance $\varepsilon \in (0, \infty)$:

(4.1)
$$|-\delta_\mathcal{K}^*(-y) - \delta_\mathcal{C}^*(-z) - \langle c, x \rangle| \leq \varepsilon \left(1 + |\delta_\mathcal{K}^*(-y) + \delta_\mathcal{C}^*(-z)| + |\langle c, x \rangle|\right),$$
$$\|Ax - \Pi_\mathcal{K}(Ax)\| \leq \varepsilon \left(1 + \|\bar{b}\|\right),$$
$$\|c - A^*y - z\| \leq \varepsilon \left(1 + \|c\|\right),$$

where $\bar{b} := \max(|l_c|, |u_c|)$ is taken componentwise, treating any infinite entries in l_c or u_c as zero when computing the norm. Other solvers are run with their default

---

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

stopping rules, which are comparable to (4.1). In Subsection 4.2, we refer to the relative KKT residual and duality gap in (4.1) collectively as the *Optimality* to assess the effect of algorithmic enhancements.

4.2. Impact of algorithmic enhancements. We evaluate the impact of adaptive restarts, penalty parameter updates, and the relaxation step on two representative Mittelmann LP instances (`datt256` and `nug08-3rd`). We consider these components incrementally: starting from the baseline PR method (Algorithm 2.1 without the Halpern step), then HPR (Algorithm 2.1), HPR with adaptive restarts, HPR with adaptive restarts plus adaptive penalties (Algorithm 2.2), and finally an HDR variant (obtained by replacing Step 4 in Algorithm 2.1 with $\hat{w}^{k+1} = \bar{w}^{k+1}$), combined with the similar restart and adaptive penalty updates as in HPR-LP (Algorithm 2.2). The results are reported in Fig. 4.1, and the main observations are as follows:

1. The plain PR method fails to converge. In contrast, HPR without restart converges, albeit slowly. Incorporating adaptive restarts upgrades the rate to linear convergence, and adaptive penalty updates further accelerate convergence.

2. When both restart and adaptive penalty updates are employed, HPR consistently outperforms HDR, underscoring the critical role of the relaxation step (Step 4 in Algorithm 2.1) in enhancing performance.
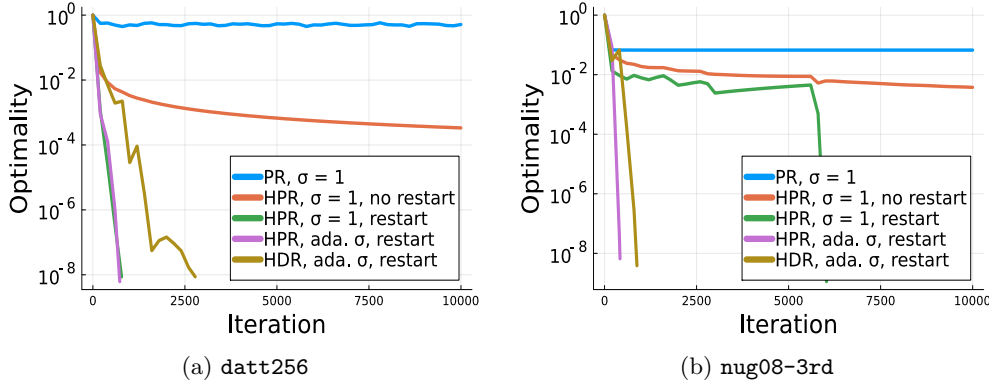

(a) `datt256`   (b) `nug08-3rd`

Fig. 4.1: Performance of HPR methods with different enhancements.

Restart. We next compare the adaptive restart strategy with fixed-frequency restart. Fig. 4.2 reports the performance of the HPR method in Algorithm 2.1 (with σ fixed at 1) under three settings: no restart, fixed-frequency restart (with the frequency tuned for each instance), and adaptive restart as described in Subsection 2.2. When a restart strategy is applied, the HPR method exhibits linear convergence, with the rate depending on the restart schedule. Overall, adaptive restart consistently achieves slightly better performance than the best fixed-frequency choice.
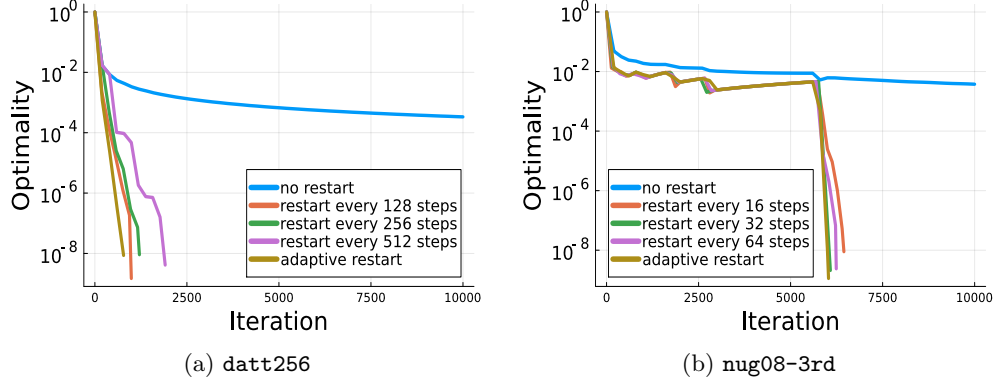
Fig. 4.2: Performance of HPR methods under different restart strategies.

Penalty parameter σ. We next evaluate the effect of the penalty parameter in the HPR method with adaptive restart. Fig. 4.3 reports the performance of HPR with different fixed values of σ as well as with adaptive updates. We observe that for certain fixed choices of σ (e.g., $\sigma = 100$ for `datt256` and $\sigma = 0.1$ or 1 for `nug08-3rd`), the HPR method often exhibits a two-phase behavior: an initial slow progress phase (phase I), followed by linear convergence (phase II). The choice of σ significantly affects the length of phase I, while adaptive σ selection consistently shortens this phase and yields the best overall performance.
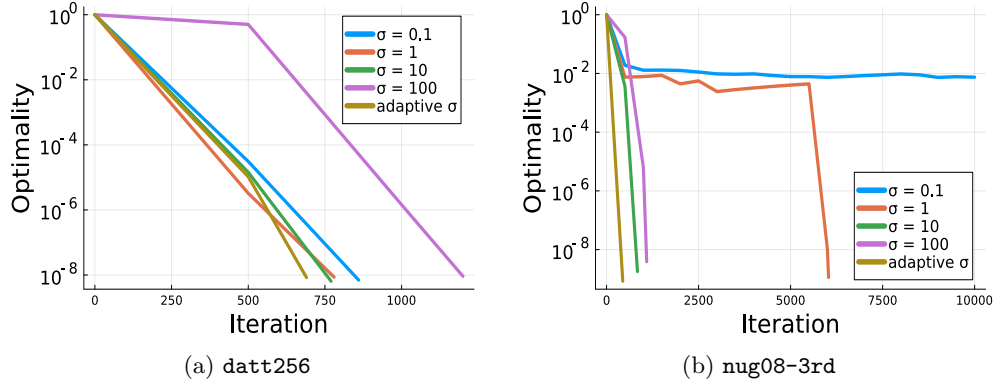
Fig. 4.3: Performance of HPR methods under different penalty parameter strategies.

4.3. Performance comparison on benchmark datasets. We compare the performance of several state-of-the-art solvers on two benchmark datasets, including 49 instances from the Mittelmann LP benchmark and a subset of 18 large-scale instances from MIPLIB 2017. The tested solvers include HPR-LP (Julia implementation)[3] [8], EPR-LP (Julia implementation, incorporating restart and penalty pa-

---

[3]https://github.com/PolyU-IOR/HPR-LP, v0.1.2

rameter update techniques similar to HPR-LP) [10], cuPDLPx (C implementation)[4] [34], and the GPU-accelerated PDLP family: cuPDLP.jl (Julia implementation)[5] [35], cuPDLP-C (C implementation)[6] [39], and NVIDIA cuOpt (C implementation)[7]. For comparison with the commercial solver, we also include Gurobi[8] [22], run with its barrier method. To ensure fairness, the presolve option is disabled for all solvers.

We first compare two implementations of the HPR method for LP (Algorithm 2.1): HPR-LP and cuPDLPx. The results on the Mittelmann LP benchmark and the MIPLIB 2017 relaxations are reported in Tables 4.1 and 4.2, respectively. On the Mittelmann benchmark, cuPDLPx performs comparably to HPR-LP, with both solving 44/49 instances, while HPR-LP is about $1.1\times$ faster in terms of SGM10. On the MIPLIB relaxations, HPR-LP shows a clearer advantage, solving two additional instances and achieving a $\mathbf{1.8\times}$ speedup. These results confirm that both solvers are effective realizations of the HPR method, with our HPR-LP implementation exhibiting greater robustness and efficiency. Therefore, we do not include cuPDLPx in subsequent comparisons with other solvers.

Table 4.1: Comparison between HPR-LP and cuPDLPx on the Mittelmann LP benchmark (49 instances). Accuracy $= 10^{-8}$, time limit 15,000s.

Solvers	HPR-LP	cuPDLPx
SGM10 (s)	82.1	90.7
Solved	44	44

Table 4.2: Comparison between HPR-LP and cuPDLPx on MIPLIB 2017 LP relaxations (18 instances). Accuracy $= 10^{-8}$, time limit 18,000s.

Solvers	HPR-LP	cuPDLPx
SGM10 (s)	204.2	368.0
Solved	17	15

Table 4.3 summarizes the SGM10 runtimes and number of solved instances for HPR-LP and other tested solvers on the Mittelmann LP benchmark (49 instances). Detailed runtimes for each instance are provided in Table 5.1. Specifically, HPR-LP delivers the best overall performance, attaining both the lowest SGM10 and the largest number of solved instances. EPR-LP is slower than HPR-LP but still outperforms all GPU-accelerated PDLP variants. In terms of speedup, HPR-LP is approximately $\mathbf{3.4\times}$ faster than cuPDLP.jl, $\mathbf{2.1\times}$ faster than cuPDLP-C, $\mathbf{2.1\times}$ faster than NVIDIA cuOpt, and $\mathbf{1.4\times}$ faster than Gurobi. Moreover, HPR-LP solves more instances than cuOpt and Gurobi, underscoring both its efficiency and robustness on large-scale LP problems.

---

[4]https://github.com/MIT-Lu-Lab/cuPDLPx, v0.1.0
[5]https://github.com/jinwen-yang/cuPDLP.jl, downloaded on July 24, 2024
[6]https://github.com/COPT-Public/cuPDLP-C, v0.4.1
[7]https://github.com/NVIDIA/cuopt, v25.08.00
[8]Version 12.0.2, academic license

Table 4.3: Performance summary of solvers on the Mittelmann LP benchmark (49 instances). Accuracy 10^{-8}, time limit 15,000s.

Solvers	HPR-LP	EPR-LP	cuPDLP.jl	cuPDLP-C	cuOpt	Gurobi
SGM10 (s)	82.1	109.8	277.9	176.5	174.5	119.0
Solved	44	43	40	40	40	41

Table 4.4 reports the SGM10 runtimes and the number of solved instances on 18 large-scale MIP relaxations from MIPLIB 2017. Detailed runtimes for each instance are provided in Table 5.2. Specifically, HPR-LP achieves the best overall performance, with the lowest SGM10 and the largest number of solved instances (17/18). EPR-LP ranks second: it is slower than HPR-LP but still faster than all PDLP variants and Gurobi. In terms of speedup, HPR-LP is approximately **2.1**× faster than cuPDLP.jl, **1.6**× faster than cuPDLP-C, **1.4**× faster than NVIDIA cuOpt, and **1.9**× faster than Gurobi. Moreover, HPR-LP solves three more instances than cuPDLP.jl, cuPDLP-C, and Gurobi (17 vs. 14), demonstrating its robustness on large-scale LP relaxations of MIPs.

Table 4.4: Performance summary of solvers on 18 MIP relaxations from MIPLIB 2017. Accuracy 10^{-8}, time limit 18,000s.

Solvers	HPR-LP	EPR-LP	cuPDLP.jl	cuPDLP-C	cuOpt	Gurobi
SGM10 (s)	204.2	289.4	428.6	321.6	294.9	396.1
Solved	17	16	14	14	16	14

Overall, the numerical experiments demonstrate that HPR-LP consistently achieves the best performance across different benchmark datasets. It not only attains the lowest SGM10 runtimes but also solves the largest number of instances, confirming both its efficiency and robustness.

5. Conclusions and future directions. In this paper, we studied the relationships among GPU-accelerated FOMs for LP, focusing on HPR-LP and its links to other recent solvers. Specifically, we showed that the base algorithm of cuPDLPx is a special case of the base algorithm of HPR-LP and proved that, once active sets are identified, HPR-LP and EPR-LP become equivalent through the correspondence between Halpern iteration and ergodic iteration. Numerical experiments on benchmark datasets confirmed the efficiency and robustness of HPR-LP. These results motivate the HPR framework as a solid foundation for advancing GPU-accelerated solvers for LP and related optimization problems. Several promising directions remain for developing the next generation of high-performance solvers based on HPR methods for LP and beyond:

1. First, recent progress highlights the broader potential of HPR methods: Chen et al. [9] proposed a dual HPR method for CCQP, termed HPR-QP, which significantly outperforms existing solvers such as PDQP [38] and SCS [42, 41]. Building on the connections established in this paper, one may consider combining the implementation strategies of different solvers to produce more efficient realizations of the HPR method for solving large-scale LP problems and beyond.

2. Second, an important theoretical direction is to establish linear convergence guarantees for restarted variants of the HPR method, as empirically observed in Fig. 4.2. In [36], Lu and Yang proved that the restarted r^2HPDHG method with $\eta < 1/\|A\|$ achieves a linear rate at the restart points:

$$\text{dist}_{\mathcal{P}_\eta}\left(u^{r+1,0}, \mathcal{U}^*\right) \leq \left(\tfrac{1}{e}\right)^{r+1} \text{dist}_{\mathcal{P}_\eta}\left(u^{0,0}, \mathcal{U}^*\right) \quad \forall r \geq 0,$$

where $\mathcal{U}^*$ denotes the optimal solution set, and

$$\mathcal{P}_\eta = \begin{pmatrix} \frac{1}{\eta}I_m & A \\ A^* & \frac{1}{\eta}I_n \end{pmatrix}$$

is positive definite whenever $\eta < 1/\|A\|$. At the critical parameter $\eta = 1/\|A\|$, however, $\mathcal{P}_\eta$ becomes merely positive semidefinite, so $\text{dist}_{\mathcal{P}_\eta}$ is no longer a valid distance and the above linear rate guarantee breaks down. Given the equivalence between the base algorithm of HPR-LP and cuPDLPx (r^2HPDHG) in Proposition 3.1, it would be interesting to combine this equivalence with the theoretical framework of HPR developed in [47] to investigate whether linear convergence guarantees can be extended to the degenerate case $\eta = 1/\|A\|$.

3. Third, on the practical side, integrating AI and machine learning into HPR methods—for instance, using reinforcement learning to design restart rules or adapt penalty parameters [27]—offers a promising avenue to further enhance their robustness and efficiency.

REFERENCES

[1] 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. *Advances in Neural Information Processing Systems*, 34:20243–20257, 2021.

[2] D. Applegate, M. Díaz, O. Hinder, H. Lu, M. Lubin, B. O'Donoghue, and W. Schudy. PDLP: A practical first-order method for large-scale linear programming. *arXiv preprint arXiv:2501.07018*, 2025.

[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] K. Basu, A. Ghoting, R. Mazumder, and Y. Pan. ECLIPSE: An extreme-scale linear program solver for web-applications. In *International Conference on Machine Learning*, pages 704–714. PMLR, 2020.

[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] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vision*, 40:120–145, 2011.

[7] A. Chambolle and T. Pock. On the ergodic convergence rates of a first-order primal–dual algorithm. *Math. Program.*, 159(1):253–287, 2016.

[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. Math. Program. Comput. (2025) XXX, in print.

[9] K. Chen, D. F. Sun, Y. Yuan, G. Zhang, and X. Zhao. HPR-QP: A dual Halpern Peaceman–Rachford method for solving large-scale convex composite quadratic programming. *arXiv preprint arXiv:2507.02470*, 2025.

[10] K. Chen, D. F. Sun, Y. Yuan, G. Zhang, and X. Zhao. Peaceman–Rachford splitting method converges ergodically for solving convex optimization problems. *arXiv preprint arXiv:2501.07807*, 2025.

[11] 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.

[12] Y. Cui, X. Li, D. F. Sun, and K.-C. Toh. 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.*, 169:1013–1041, 2016.

[13] G. B. Dantzig. *Linear Programming and Extensions.* Princeton university press, 1963.

[14] D. Davis and W. Yin. Convergence rate analysis of several splitting schemes. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 115–163. Springer, 2016.

[15] Q. Deng, Q. Feng, W. Gao, D. Ge, B. Jiang, Y. Jiang, J. Liu, T. Liu, C. Xue, Y. Ye, and C. Zhang. An enhanced alternating direction method of multipliers-based interior point method for linear and conic optimization. *INFORMS J. Comput*, 37(2):338–359, 2025.

[16] 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.

[17] E. Esser, X. Zhang, and T. F. Chan. 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.

[18] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Comput. Math. Appl.*, 2(1):17–40, 1976.

[19] A. Gleixner, G. Hendel, G. Gamrath, T. Achterberg, M. Bastubbe, T. Berthold, P. Christophel, K. Jarck, T. Koch, J. Linderoth, et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Math. Program. Comput.*, 13(3):443–490, 2021.

[20] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue Française D'automatique, Informatique, Recherche Opérationnelle. Analyse Numérique*, 9(R2):41–76, 1975.

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

[22] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2025.

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

[24] 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.

[25] 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.

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

[27] J. Ichnowski, P. Jain, B. Stellato, G. Banjac, M. Luo, F. Borrelli, J. E. Gonzalez, I. Stoica, and K. Goldberg. Accelerating quadratic optimization with reinforcement learning. *Advances in Neural Information Processing Systems*, 34:21043–21055, 2021.

[28] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[29] D. Körnlein. Quantitative results for Halpern iterations of nonexpansive mappings. *J. Math. Anal. Appl.*, 428(2):1161–1172, 2015.

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

[31] 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.

[32] 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.

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

[34] H. Lu, Z. Peng, and J. Yang. cuPDLPx: A further enhanced GPU-based first-order solver for linear programming. *arXiv preprint arXiv:2507.14051*, 2025.

[35] 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.

[36] H. Lu and J. Yang. Restarted Halpern PDHG for linear programming. *arXiv preprint arXiv:2407.16144*, 2024.

[37] H. Lu and J. Yang. An overview of GPU-based first-order methods for linear programming and extensions. *arXiv preprint arXiv:2506.02174*, 2025.

[38] H. Lu and J. Yang. A practical and optimal first-order method for large-scale convex quadratic programming. *Math. Program.*, pages 1–38, 2025.

[39] 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.

[40] R. D. Monteiro and B. F. Svaiter. Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. *SIAM J. Optim.*, 23(1):475–507, 2013.

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

[42] 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.

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

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

[45] L. Shen and S. Pan. Weighted iteration complexity of the sPADMM on the KKT residuals for convex composite optimization. *arXiv preprint arXiv:1611.03167*, 2016.

[46] 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.

[47] 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.

[48] R. Wittmann. Approximation of fixed points of nonexpansive mappings. *Arch. Math.*, 58(5):486–491, 1992.

[49] M. Xu and T. Wu. A class of linearized proximal alternating direction methods. *J. Optim. Theory Appl.*, 151:321–337, 2011.

[50] 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.

[51] G. Zhang, K. Chen, Y. Yuan, X. Zhao, and D. F. Sun. On the ergodic convergence properties of the Peaceman–Rachford method and their applications in solving linear programming. *OpenReview preprint*, 2024.

[52] 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.*, 47(8):6703–6714, 2025.

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

[54] M. Zhu and T. Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *UCLA Cam Report*, 34(2), 2008.

Appendix. In this appendix, we provide detailed solver results for the Mittelmann LP benchmark (49 instances) and for the MIP relaxations (18 instances) from MIPLIB 2017, reported in Table 5.1 and Table 5.2, respectively.

Table 5.1: Results on the Mittelmann LP benchmark (49 instances). Accuracy 10^{-8}, time limit 15,000s. "T" indicates reaching the time limit, "M" indicates out-of-memory, and "F" indicates solver failure.

Instance	HPR-LP	EPR-LP	cuPDLP.jl	cuPDLP-C	cuOpt	Gurobi
a2864	5.0	5.8	2.7	1.3	1.3	F
bdry2	3950.2	9367.0	T	T	T	10.7
cont1	56.2	92.2	1871.5	389.2	361.0	1.0
cont11	429.4	282.0	5373.9	1947.8	968.0	4.2
datt256_lp	0.4	0.4	2.1	0.2	0.2	1.1
degme	40.2	45.9	85.8	60.1	64.7	17.4
dlr1	12947.4	T	T	T	T	411.6
dlr2	T	T	T	T	T	7103.3
Dual2_5000	72.8	54.7	9441.1	3310.1	747.6	M
ex10	0.5	0.5	1.5	0.1	0.2	F
fhnw-bin1	106.3	307.2	526.7	74.0	244.2	101.1
FOME13	1.5	3.3	20.3	5.4	3.5	3.5
graph40-40_lp	0.4	0.3	1.5	0.1	0.2	24.9
irish-electricity	T	T	T	T	T	F
L1_sixm1000obs	13.9	20.4	38.6	37.5	74.4	M
L1_sixm250obs	2.5	5.4	11.3	11.9	9.7	228.6
L2CTA3D	43.3	49.5	16.3	9.3	12.0	542.1
Linf_five20c	3.7	30.8	T	T	T	10.7
neos	27.9	83.9	396.0	199.4	162.5	212.0
neos-3025225_lp	5.4	11.5	9.7	3.9	3.3	24.1
neos-5052403-cygnet	10.6	10.2	16.3	8.6	6.7	11.3
neos-5251015_lp	1.1	3.8	3.4	1.2	1.2	5.7
neos3	0.4	0.3	2.0	0.6	0.4	321.3
ns168703	T	T	T	T	T	4.4
ns168892	T	T	T	T	T	F
nug08-3rd	0.1	0.2	1.7	0.1	0.1	4.8
pds-100	3.0	4.5	21.2	8.7	7.8	19.5
physiciansched3-3	T	T	T	T	T	89.3
Primal2_1000	214.2	402.9	2105.8	1556.6	1069.0	F
qap15	1.8	2.7	14.3	5.4	2.9	1.2
rail02	47.6	75.6	852.4	145.6	168.0	9.8
rail4284	137.3	397.6	578.7	362.4	482.1	13.2
rmine15_lp	3.5	6.3	24.3	10.0	9.8	156.3
s100	148.5	319.8	1631.4	571.4	499.4	6.7
s250r10	18.3	37.1	192.6	105.4	35.6	3.4
s82	1157.1	3803.6	3390.7	4686.7	6204.9	75.0
savsched1	0.2	0.3	1.9	0.3	0.2	6.1
scpm1_lp	31.7	53.4	41.9	27.5	19.6	9.2
set-cover	287.5	383.7	515.3	290.1	385.5	60.5
shs1023	110.5	144.5	T	T	T	32.8
square41	384.3	326.6	234.9	138.3	89.3	79.5
stat96v2	459.9	1234.5	1705.9	1290.1	9740.8	F
stormG2_1000	10.6	11.9	41.0	30.5	30.5	23.1
stp3d	15.6	24.9	157.0	34.4	27.4	5.3
supportcase10	5.3	6.1	20.9	6.9	5.2	19.5
thk_48	572.2	489.3	2839.6	1164.5	755.9	7581.2
thk_63	28.0	47.8	101.4	49.2	54.3	270.5
tpl-tub-ws1617	38.6	66.2	554.3	151.1	220.2	51.7
woodlands09	0.7	0.6	2.3	0.7	0.5	7.4
SGM10	82.1	109.8	277.9	176.5	174.5	119.0
solved	44	43	40	40	40	41

Table 5.2: Results on MIP relaxations (18 instances) from MIPLIB 2017. Accuracy 10^{-8}, time limit 18,000s. "T" indicates reaching the time limit, "M" indicates out-of-memory, and "F" indicates solver failure.

Instance	HPR-LP	EPR-LP	cuPDLP.jl	cuPDLP-C	cuOpt	Gurobi
a2864-99blp	4.9	5.8	8.1	1.3	1.3	F
dlr1	12934.8	T	T	T	T	412.6
ivu06-big	624.5	905.3	4203.3	763.7	1347.3	24.8
ivu59	929.3	1061.7	3500.5	368.3	401.7	30.9
kottenpark09	80.4	52.0	60.9	74.6	59.5	278.2
neos-2991472-kalu	0.4	0.9	0.5	0.1	0.1	91.1
neos-3208254-reiu	35.3	72.3	43.0	116.0	170.0	640.6
neos-3740487-motru	1.5	2.4	1.6	3.4	3.1	M
neos-4332801-seret	301.0	1229.0	T	T	9260.0	419.2
neos-4332810-sesia	272.8	785.0	T	T	2575.2	F
neos-4535459-waipa	T	T	T	T	T	F
neos-4545615-waita	297.6	394.1	393.4	2915.2	3069.1	39.3
nucorsav	3.2	5.6	3.5	4.9	3.7	1862.0
scpn2	173.5	121.5	321.8	44.6	37.8	17.5
square41	383.0	327.7	328.0	138.0	89.6	79.0
square47	548.1	697.7	571.8	133.8	153.8	57.3
t11nonreg	1498.1	1683.9	1821.7	2136.0	2514.5	104.2
usafa	30.6	318.0	33.3	33.2	66.5	90.6
SGM10	204.2	289.4	428.6	321.6	294.9	396.1
solved	17	16	14	14	16	14