

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

**FAST ALGORITHMS FOR OPTIMAL
TRANSPORT PROBLEMS: THEORY,
ACCELERATION, AND EXTENSIONS**

ZHANG GUOJUN

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University

Department of Applied Mathematics

Fast Algorithms for Optimal Transport Problems:
Theory, Acceleration, and Extensions

Zhang Guojun

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

August 2025

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

Zhang Guojun

(Name of student)

To my parents

Abstract

Optimal Transport (OT) has become a powerful tool for comparing probability distributions, but solving OT-related problems remains computationally intensive. In the discrete setting, many important OT tasks—such as computing the Kantorovich–Wasserstein (KW) distance and Wasserstein barycenters with pre-specified finite supports—can be formulated as large-scale linear programs, which form a special class of linearly constrained convex optimization problems (COPs). This thesis develops scalable, accelerated algorithms based on the Alternating Direction Method of Multipliers (ADMM) to efficiently solve COPs, with applications to these key OT problems as well as extensions to general linear programming (LP).

First, we propose a Halpern–Peaceman–Rachford (HPR) algorithm without proximal terms for solving COPs by applying Halpern iteration to the Peaceman–Rachford splitting method. The HPR algorithm achieves a non-ergodic iteration complexity of $O(1/\varepsilon)$ for obtaining an ε -approximate solution in terms of the Karush–Kuhn–Tucker (KKT) residual and the objective error. When applied to the Wasserstein barycenter problem with pre-specified finite supports—that is, computing the barycenter of T discrete probability distributions, each supported on m points—we derive closed-form solutions for the subproblems, leading to an overall complexity of $O(Tm^2/\varepsilon)$ flops. This improves upon the previous best-known bound of $\tilde{O}(Tm^2/\varepsilon)$ by eliminating extraneous logarithmic factors.

Second, we develop an accelerated preconditioned ADMM (pADMM) method

with semi-proximal terms for solving COPs, where the HPR algorithm arises as a special case when the proximal terms are removed. To accelerate pADMM, we reformulate it as a proximal point algorithm (PPA) equipped with a positive semidefinite, possibly degenerate, preconditioner, due to the lack of strong convexity of the proximal terms in the pADMM. By integrating Halpern iteration and fast Krasnosel’skiĭ–Mann iteration into the resulting degenerate PPA (dPPA), we obtain asymptotic $o(1/k)$ and non-asymptotic $O(1/k)$ iteration complexity. Building on this, we develop an accelerated pADMM algorithm that achieves these rates for the KKT residual and the objective error. We further apply this framework to develop HOT, a Halpern accelerating algorithm for computing the KW distance. Instead of solving the original OT problem directly, HOT solves an equivalent reduced formulation that leverages the problem structure to reduce computational complexity. For the discrete OT problem with m support points in $\mathbb{R}^2$, under an L_2^2 -ground cost, we derive closed-form solutions for HOT’s subproblems, resulting in an overall computational complexity of $O(m^{1.5}/\varepsilon)$ flops—surpassing the previous best-known bound of $\tilde{O}(m^2/\varepsilon)$.

Finally, inspired by the promising performance of the HPR method for solving OT-related problems, we introduce HPR-LP, an implementation of the HPR method with semi-proximal terms for solving large-scale general LP problems. Based on the $O(1/k)$ complexity results of the HPR method, 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 an NVIDIA A100-SXM4-80GB GPU under different stopping tolerances. The Julia implementation of HPR-LP 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} .

Acknowledgements

I would like to express my deepest gratitude to my advisor, **Professor Sun Defeng**, for his patient guidance, generous support, and unwavering dedication throughout my doctoral studies. He was always thoughtful and encouraging—offering insight even when it meant missing his last MTR home or sacrificing his sleep. He introduced me to the field of optimization, guided me through foundational texts such as Convex Analysis, and inspired me with stories of pioneers like the Dantzig Prize winners. Beyond academic training, Professor Sun patiently helped me improve my English pronunciation—often reading words aloud for me—and guided me in becoming a more articulate communicator. More importantly, he led by example, encouraging me to grow into a more disciplined, thoughtful, and well-rounded person. His academic integrity, kindness, and passion for research have left a lasting impression on me, shaping both my intellectual development and personal growth. I owe him a great debt of gratitude.

Special thanks go to my collaborators — Yuan Yancheng, Zhao Xinyuan, Chen Kaihuang, and Gu Zhexuan — for their insightful discussions and shared experiences throughout this journey.

My heartfelt appreciation also goes to both former and current members of the optimization group, especially Zhang Min, Chen Liang, Wu Can, Li Qian, Ma Chiyu, and Yan Hui, for their camaraderie, encouragement, and support along the way.

I am grateful to all my friends in Hong Kong, in particular Liu Yan, Wang Qi, Li

Jiahong, Chen Ruoning, and Zhao Wei, for their friendship, memorable gatherings, and countless conversations that brought joy and balance to my life.

I would also like to thank the faculty and staff of the Department of Applied Mathematics at The Hong Kong Polytechnic University for fostering a vibrant research environment, offering generous support for conference travel, and providing excellent research facilities.

Finally, I owe my deepest thanks to my family for their unconditional love, unwavering encouragement, and constant understanding — without which this thesis would not have been possible.

Contents

Certificate of Originality	iii
Abstract	ix
Acknowledgements	xi
List of Figures	xvii
List of Tables	xix
1 Introduction	1
1.1 Motivations and Related Work	1
1.1.1 Kantorovich–Wasserstein Distance	2
1.1.2 Wasserstein Barycenter Problem	6
1.2 Contributions	10
1.3 Thesis Organization	12
2 Preliminaries	15
2.1 Notation	15
2.2 Degenerate Proximal Point Algorithm	16
2.3 Fixed-point Iteration Methods	20
2.4 Preconditioned (Semi-proximal) ADMM	22
3 A Halpern Peaceman–Rachford (HPR) Method for Solving Wasserstein Barycenter Problems	29
3.1 An HPR Method for Convex Optimization Problems	29

3.1.1	Convergence Analysis	37
3.1.2	Iteration Complexity Analysis	39
3.2	An HPR Method for Wasserstein Barycenter Problems	45
3.3	Experiments	54
3.3.1	Experiments on Synthetic Data	57
3.3.2	Experiments on Real Data	63
4	An Accelerated Preconditioned (Semi-proximal) ADMM Method for OT Problems	69
4.1	An Accelerated Degenerate Proximal Point Algorithm	69
4.2	Acceleration of the Preconditioned ADMM	77
4.2.1	The Equivalence Between the pADMM and the dPPA	78
4.2.2	An Accelerated pADMM	83
4.3	A Halpern Accelerating Algorithm for Solving Reduced OT Problems	94
4.3.1	An Equivalent Reduced Model of the OT Problem	94
4.3.2	Reconstruct the Transport Plan from the Reduced Model	98
4.3.3	HOT: A Halpern Accelerating Method for Solving OT Problems	100
4.4	Experiments	108
4.4.1	Numerical Comparison on the DOTmark Dataset	108
4.4.2	An Application in Color Transfer	118
5	Extension–HPR-LP: An Implementation of an HPR Method for Solving Linear Programming	121
5.1	An HPR Method for Solving General LP Problems	121
5.2	An HPR Solver for LP Problems	129
5.2.1	Restart Criteria	130
5.2.2	Update Rule for σ	131
5.2.3	The GPU Implementation of the HPR-LP	134

5.3	Numerical Experiments	135
5.3.1	Experimental Setup	135
5.3.2	Mittelmann’s LP Benchmark Set	138
5.3.3	MIP Relaxations	143
5.3.4	Large-scale Applications: QAP, ZIB Problem, and PageRank Instances	145
6	Conclusions and Future Work	151
	Bibliography	155

List of Figures

3.1	Comparisons between Gurobi, fast-ADMM, HPR, and HPR-hybrid . . .	62
3.2	Barycenters obtained by running different methods for 50s and 100s, respectively.	65
3.3	Barycenters obtained by running different methods on the Coil20 data set for the 100th, 200th, and 300th iterations.	66
3.4	Barycenters obtained by running different methods on the Yale Face B data set for the 100th, 200th, and 300th iterations.	67
4.1	A visualization of the selected images from the DOTmark Dataset is presented. The upper row features images from the Classic Images category, while the bottom row contains images from the Shapes category.	109
4.2	Comparison of solving the linear system in (4.94) using Proposition 4.7 and sparse Cholesky decomposition. The time for Cholesky-based solving is split into two parts: decomposition (orange) and forward-backward substitution (blue). For resolutions 256×256 and 512×512 , Cholesky decomposition failed due to out-of-memory errors during testing.	115
4.3	Selected examples of color transfer based on the reduced OT model with the OT plan recovered by Algorithm 9.	119
5.1	Relative impact of HPR-LP's enhancements on Mittelman's LP benchmark set without presolve.	140

List of Tables

1.1	Selected known complexity results for solving the OT problem. (C represents the largest entries of the cost matrix, while R denotes the distance between the initial point and the solution set.)	5
1.2	Selected known complexity results for solving the WBP with T sample distributions and m supports. (C represents the largest entries of the cost matrix, while R_1 denotes the distance between the initial point and the solution set, and R_2 in the complexity bound of the sGS-ADMM comes from the additional proximal term.)	9
3.1	Numerical results on Gaussian mixture distributions	59
3.2	Summary of real datasets	63
3.3	Computation times (in seconds) of different methods for computing the Wasserstein barycenter on the Coil20 and Yale B face datasets.	68
4.1	The numerical results of different algorithms on the DOTmark dataset.	112
4.2	The numerical results of the convolutional Sinkhorn method with different λ	113
4.3	Numerical results of various algorithms on the DOTmark dataset: total time = solve time + transport plan recovery time.	116
4.4	Comparison of HOT's performance on CPU and GPU.	116
4.5	Numerical results of HOT and ADMM using absolute KKT residual as the stopping criterion (10^{-6}).	117
5.1	Scales of instances in MIP relaxations.	136
5.2	Numerical performance of different solvers on 49 instances of Mittelmann's LP benchmark set with presolve.	141

5.3	Numerical performance of different solvers on 49 instances of Mittelmann’s LP benchmark set without presolve.	141
5.4	Shifted geometric mean of iteration counts for different solvers on 49 instances of Mittelmann’s LP benchmark set, with and without presolve.	142
5.5	Per-iteration time in seconds of different solvers for Mittelmann’s LP benchmark set.	143
5.6	Numerical performance of different solvers on 380 instances of MIP relaxations with presolve.	144
5.7	Numerical performance of different solvers on 380 instances of MIP relaxations without presolve.	144
5.8	Solving time in seconds and SGM10 for different solvers on 20 QAP instances (Burkard et al., 1997) with presolve.	146
5.9	Solving time in seconds and SGM10 for different solvers on 20 QAP instances (Burkard et al., 1997) without presolve.	147
5.10	Solving time in seconds for the “zib03” instance (Koch et al., 2022). .	148
5.11	Solving time in seconds and SGM10 for different solvers on 4 PageRank instances without presolve.	149

Chapter 1

Introduction

1.1 Motivations and Related Work

The Optimal Transport (OT) problem seeks the most efficient way to transport mass between two probability distributions while minimizing a prescribed cost function (Monge, 1781). This thesis focuses on the design and analysis of scalable algorithms for solving large-scale OT-related problems in the discrete setting. In particular, we study a class of discrete OT-related problems that admit a linear programming (LP) formulation of the following form:

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & \langle c, x \rangle + \delta_K(x) \\ \text{s.t.} \quad & Ax = b, \end{aligned} \tag{1.1}$$

where $A \in \mathbb{R}^{M \times N}$, $b \in \mathbb{R}^M$, and $c \in \mathbb{R}^N$. Here, $\delta_K(\cdot)$ is the indicator function over the nonnegative orthant $K := \{x \in \mathbb{R}^N \mid x \geq 0\}$, which imposes nonnegativity constraints. The dual of problem (1.1) takes the form:

$$\min_{y \in \mathbb{R}^M, z \in \mathbb{R}^N} \{ -\langle b, y \rangle + \delta_K^*(-z) \mid A^*y + z = c \}, \tag{1.2}$$

where $\delta_K^*(\cdot)$ denotes the support function of the cone K and A^* denotes the adjoint of A .

This LP formulation serves as a unifying framework for several important problems related to discrete OT. In the following sections, we focus on two representative

examples: the computation of the Kantorovich–Wasserstein (KW) distance and the Wasserstein barycenter problem (WBP) with pre-specified finite supports. Both can be cast as special cases of problem (1.1). We present an overview of their formulations, key computational challenges, and recent algorithmic developments.

1.1.1 Kantorovich–Wasserstein Distance

Consider a discrete probability distribution with a finite set of support points:

$$\mathcal{P} := \{(a_i, \mathbf{q}_i) \in \mathbb{R}_+ \times \mathbb{R}^d : i = 1, \dots, m\},$$

where $\{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ are the support points and $\mathbf{a} := (a_1, \dots, a_m)$ is the associated probability vector satisfying $\sum_{i=1}^m a_i = 1$. Given two discrete distributions $\mathcal{P}^u = \{(a_i^u, \mathbf{q}_i^u) : i = 1, \dots, m_u\}$ and $\mathcal{P}^v = \{(a_i^v, \mathbf{q}_i^v) : i = 1, \dots, m_v\}$, the p -KW distance (Kantorovich, 1942) between $\mathcal{P}^u$ and $\mathcal{P}^v$ is defined as the optimal objective value of the following OT problem:

$$\begin{aligned} (\mathcal{W}_p(\mathcal{P}^u, \mathcal{P}^v))^p &:= \min_{\pi \in \mathbb{R}^{m_u \times m_v}} \langle \pi, \mathcal{D}(\mathcal{P}^u, \mathcal{P}^v) \rangle \\ \text{s.t. } \quad &\pi^* \mathbf{1}_{m_u} = \mathbf{a}^v, \\ &\pi \mathbf{1}_{m_v} = \mathbf{a}^u, \\ &\pi \geq \mathbf{0}, \end{aligned} \tag{1.3}$$

where the distance matrix $\mathcal{D}(\mathcal{P}^u, \mathcal{P}^v) \in \mathbb{R}^{m_u \times m_v}$ is given by

$$\mathcal{D}(\mathcal{P}^u, \mathcal{P}^v)_{ij} = \|\mathbf{q}_i^u - \mathbf{q}_j^v\|_p^p, \quad \text{with } p \geq 1.$$

The KW distance has emerged as a premier metric for quantifying the similarity between two probability distributions. It has achieved considerable success across numerous fields, such as color transfer (Pitié and Kokaram, 2007; Bonneel et al., 2013; Solomon et al., 2015), texture synthesis and mixing (Dominitz and Tannenbaum, 2009), registration and warping (Haker et al., 2004), transport-based morphometry

(Basu et al., 2014), and hypothesis testing (Del Barrio et al., 1999). Despite its robust geometric foundation and wide applicability, computationally evaluating the KW distance remains expensive, particularly for large-scale problems (Villani, 2003; Peyré and Cuturi, 2019).

When it comes to computing the KW distance between two discrete probability distributions with m supports, there are mainly two popular approaches: (i) Computing an approximate KW distance by solving the optimal transport problem with an additional entropy regularization (Cuturi, 2013); (ii) Computing the KW distance via solving the corresponding LP problem (Orlin, 1988). Readers can refer to (Peyré and Cuturi, 2019) and the references therein for a more detailed discussion of the algorithms for solving OT problems. We briefly discuss the challenges of the aforementioned approaches.

Challenges with the Entropy-regularized Approach. Due to its scalability, the Sinkhorn algorithm and its improved versions have been widely adopted to compute an approximation of the KW distance in applications (Cuturi, 2013; Lin et al., 2019; Dvurechensky et al., 2018; Guminov et al., 2021). In particular, the Sinkhorn algorithm can efficiently solve the regularized OT problem when the regularization parameter is moderate (i.e., no less than 10^{-2}). However, a high-quality approximation of the KW distance is important for better performance in many applications, which requires solving the regularized OT problem with a small regularization parameter. Unfortunately, a small regularization parameter will usually cause numerical issues and slower convergence for the Sinkhorn algorithm. Some stabilized and rescaling techniques (Schmitzer, 2019) have been proposed to improve the robustness of solving the regularized OT problems, but the efficiency of the stabilized algorithms is unsatisfactory compared to the Sinkhorn algorithm.

Challenges with the LP-based Approach. Along this line, the interior point method (Pele and Werman, 2009) and the network simplex method (Goldberg et al., 1989; Gabow and Tarjan, 1991) are popular choices for obtaining solutions with high accuracy to the moderate-scale LP problem. Recently, a semismooth Newton-based inexact proximal augmented Lagrangian method (Li et al., 2020) has been proposed for solving linear programming problems, which can solve the OT problem as a special case. The semismooth Newton-based algorithm can exploit the sparsity of the solution by the generalized Jacobian and show superior numerical performance compared to Gurobi in some examples. However, these solvers are not applicable for solving very large-scale OT problems due to the high computational complexity. The urgent need to solve large-scale OT problems in applications inspires extensive research in designing first-order methods, such as the Douglas–Rachford splitting algorithm (Mai et al., 2022) and the primal-dual hybrid gradient method (PDHG) (Zhu and Chan, 2008; Esser et al., 2010; Chambolle and Pock, 2011; Applegate et al., 2021). Jambulapati et al. (Jambulapati et al., 2019) proposed an algorithm based on a dual-extrapolation algorithm to achieve an $\tilde{O}(m^2/\varepsilon)$ computational complexity bound, where m is the number of supports, for obtaining an ε -approximate solution (in terms of objective function value).¹ We summarize some known complexity results for solving the OT problem in Table 1.1. Readers can refer to (Tupitsa et al., 2022; Khamis et al., 2024) and the references therein for a more detailed discussion.

¹ Despite its better complexity bound, the empirical performance of this algorithm is less efficient than that of other algorithms, such as accelerated gradient method (Dvurechensky et al., 2018; Guminov et al., 2021).

Table 1.1: Selected known complexity results for solving the OT problem. (C represents the largest entries of the cost matrix, while R denotes the distance between the initial point and the solution set.)

Algorithm	Distance Type (p)	Time Complexity Result
Sinkhorn (Dvurechensky et al., 2018)	$p \geq 1$	$\tilde{O}(m^2 C^2 / \varepsilon^2)$
APDAGD (Dvurechensky et al., 2018; Lin et al., 2022)	$p \geq 1$	$\tilde{O}(m^{2.5} C / \varepsilon)$
Greenkhorn (Lin et al., 2022)	$p \geq 1$	$\tilde{O}(m^2 C^2 / \varepsilon^2)$
Accelerated Sinkhorn (Lin et al., 2022)	$p \geq 1$	$\tilde{O}(m^{7/3} C^{4/3} / \varepsilon^{4/3})$
AAM (Guminov et al., 2021)	$p \geq 1$	$\tilde{O}(m^{2.5} C / \varepsilon)$
Dual Extrapolation (Jambulapati et al., 2019)	$p \geq 1$	$\tilde{O}(m^2 C / \varepsilon)$
HPD (Chambolle and Contreras, 2022)	$p \geq 1$	$\tilde{O}(m^{2.5} C / \varepsilon)$
HPR (This thesis)	$p \geq 1$	$O(m^2 R / \varepsilon)$
HOT (This thesis)	$p = 2$	$O(m^{1.5} R / \varepsilon)$

Memory Bottlenecks and Reduced OT Models. Beyond the challenges above in computational efficiency, all these algorithms for the original OT problem with m supports require at least a memory cost of $O(m^2)$. This memory cost makes it forbidden to compute the KW distance of two distributions with massive supports (i.e., the OT problem for computing the KW distance of two 512×512 grey images has more than 6.8×10^{10} variables). To address this issue, researchers have proposed some approximate models for the OT problem, such as the linear OT framework (Wang et al., 2013), the sliced Wasserstein distance (Bonneel et al., 2011), and the approximate earth mover’s distance (Indyk and Thaper, 2003; Shirdhonkar and Jacobs, 2008; Leeb and Coifman, 2016). While these approaches alleviate memory issues and simplify computations, they inevitably introduce a loss of exactness. This limitation has motivated further research into equivalent reduced models for the OT problem. When the ground distances between supports in $\mathbb{R}^d$ are calculated by L_1 -norm, Ling and Okada proposed an equivalent reduced model with $O(dm)$ memory cost to calculate the earth mover’s distance (equivalent to the KW distance) (Ling and Okada, 2007).

Recently, Auricchio et al. (Auricchio et al., 2018) extended the idea to the case where the ground distances between supports are calculated by L_2^2 -norm and proposed an equivalent reduced model with $O(dm^{\frac{d+1}{d}})$ memory cost. The authors in (Auricchio et al., 2018) adopted the Network Simplex method to solve the reduced model and demonstrated superior performance in terms of computational and memory efficiency compared to the Sinkhorn and the convolutional Sinkhorn method (Solomon et al., 2015; Solomon, 2018) on moderately sized examples. Unfortunately, the efficiency becomes unsatisfactory for very large-scale problems. Moreover, the transport plan is not available if we solve the reduced model, which is critical in a wide range of applications, such as color transfer (Pitié and Kokaram, 2007; Ferradans et al., 2014), texture synthesis (Dominitz and Tannenbaum, 2009), and domain adaptation (Courty et al., 2016).

1.1.2 Wasserstein Barycenter Problem

The concept of the Wasserstein barycenter, introduced by Agueh and Carlier (Agueh and Carlier, 2011), generalizes the notion of the Euclidean mean to the space of probability distributions endowed with the KW distance. Given a collection of discrete probability distributions $\{\mathcal{P}^t\}_{t=1}^T$, where

$$\mathcal{P}^t = \{(a_i^t, \mathbf{q}_i^t) : i = 1, \dots, m_t\},$$

a p -Wasserstein barycenter $\mathcal{P}^c := \{(a_i^c, \mathbf{q}_i^c) : i = 1, \dots, m\}$ with m support points is a minimizer of the following optimization problem:

$$\min_{\mathcal{P}^c \in \Xi^p(\mathbb{R}^d)} \sum_{t=1}^T \omega_t (\mathcal{W}_p(\mathcal{P}^c, \mathcal{P}^t))^p, \quad (1.4)$$

where $\Xi^p(\mathbb{R}^d)$ denotes the set of all discrete probability distributions on $\mathbb{R}^d$ with finite p -th moment, and the weight vector $(\omega_1, \dots, \omega_T)$ satisfies $\sum_{t=1}^T \omega_t = 1$ and

$\omega_t > 0, t = 1, \dots, T$. Problem (1.4) is a non-convex multi-marginal OT problem since it requires optimizing both the barycenter weights $\mathbf{a}^c = (a_1^c, \dots, a_m^c)$ and the support locations $\mathcal{Q}^c := \{\mathbf{q}_i^c\}_{i=1}^m$. In many practical applications, however, the barycenter supports $\mathcal{Q}^c$ are fixed empirically. As a result, one only needs to compute the optimal weight vector $\mathbf{a}^c$ via solving the optimization problem (1.4) with finite specified supports. In this thesis, we focus on the WBP with specified supports. From now on, we assume that the support $\mathcal{Q}^c$ is given. Under this assumption, the problem reduces to solving for $\mathbf{a}^c$ via the following LP:

$$\begin{aligned} \min_{\mathbf{a}^c \in \mathbb{R}^m, \{\pi^t\}_{t=1}^T \in \mathbb{R}^{m \times m_t}} \quad & \sum_{t=1}^T \langle D^t, \pi^t \rangle \\ \text{s.t.} \quad & (\pi^t)^* \mathbf{1}_m = \mathbf{a}^t, \quad t = 1, \dots, T, \\ & \pi^t \mathbf{1}_{m_t} = \mathbf{a}^c, \quad t = 1, \dots, T, \\ & \pi^t \geq 0, \quad t = 1, \dots, T, \\ & \langle \mathbf{a}^c, \mathbf{1}_m \rangle = 1, \end{aligned} \tag{1.5}$$

where $D^t := \omega_t \mathcal{D}(\mathcal{P}^c, \mathcal{P}^t)$.

The WBP has made a significant impact in a variety of fields, including machine learning (Li and Wang, 2008; Ye and Li, 2014; Peyré and Cuturi, 2019), physics (Cotar et al., 2013), statistics (Bigot and Klein, 2018), economics (Chiappori et al., 2010; Galichon, 2016), brain imaging (Gramfort et al., 2015), and so on. However, computing the Wasserstein barycenters is computationally expensive, especially for modern applications with an immense amount of data.

Similar to the OT problem, there are two principal approaches to solving the WBP: (i) Entropy-regularized Approach: Computing the Wasserstein barycenter by solving a regularized WBP with an additional entropy term (Cuturi and Doucet, 2014; Benamou et al., 2015). (ii) LP-based Approach: Solving the WBP directly through its LP formulation. Below, we discuss the challenges associated with these approaches.

Challenges with the Entropy-regularized Approach. To overcome the computational challenges, Cuturi and Doucet (2014) applied the entropic regularization idea to study the WBP. Later, Benamou et al. (2015) proposed an iterative Bregman projection (IBP) method, which generalized Sinkhorn’s algorithm, to solve the Entropy-regularized WBP. Along this direction, numerous algorithms have been proposed, including gradient-type methods (Cuturi and Peyré, 2016), accelerated primal-dual gradient descent (Dvurechenskii et al., 2018; Krawtschenko et al., 2020), the fast IBP (Lin et al., 2020), stochastic gradient descent (Claici et al., 2018; Tiapkin et al., 2022), distributed and parallel gradient descent (Staib et al., 2017; Uribe et al., 2018; Rogozin et al., 2021). When the regularization parameter in the entropy-regularized WBP is moderate, for example, no less than 10^{-2} , the aforementioned algorithms, such as the IBP, can be effective in finding an approximate solution to the entropy-regularized WBP. However, it is known that one needs to choose a small regularization parameter in the regularized WBP to obtain a high-quality solution in real applications. A small regularization parameter will usually cause numerical issues and slower convergence for these algorithms, such as the IBP. Some stabilized and scaling techniques have been proposed to improve the robustness of the entropic-regularization-based algorithms (Schmitzer, 2019), but the efficiency of the stabilized algorithms remains unsatisfactory for solving the large-scale WBP. These issues may restrict the role of the entropic regularization approach in applications with high requirements of the solution quality (Bonneel et al., 2016).

Challenges with the LP-based Approach. The instability and unsatisfactory efficiency of the entropic-regularized algorithm for solving the WBP in the relatively high accuracy regime motivate researchers to move back to designing efficient algorithms for solving the WBP without entropic regularization. In this direction, the first-order splitting algorithms, in particular, the alternating direction method of

multipliers (ADMM) (Glowinski and Marroco, 1975; Gabay and Mercier, 1976) and its variants are very popular. When applying the ADMM directly to solve the WBP, in each iteration, one needs to solve a huge-scale linear system, which is computationally challenging or even prohibitive. To avoid solving this huge linear system, Ye et al. (2017) proposed a modified Bregman ADMM (mBADMM), where each step of the mBADMM has a closed-form solution for solving the WBP. Later, Yang et al. (2021) proposed a symmetric Gauss-Seidel ADMM (sGS-ADMM) for solving the WBP, which is more efficient and robust than the mBADMM. Regarding the WBP as a multi-block problem and partially using the special structure of the WBP, the sGS-ADMM enjoys a low computational complexity in each iteration. It has also been demonstrated in (Yang et al., 2021) that the sGS-ADMM is more stable compared to the algorithms based on entropic regularization, such as IBP. Note that the semi-proximal ADMM, including sGS-ADMM (Yang et al., 2021), has the non-ergodic $O(1/\sqrt{k})$ iteration complexity in terms of the Karush-Kuhn-Tucker (KKT) residual (Cui et al., 2016). It would be interesting to design accelerated ADMM-type methods with better $O(1/k)$ iteration complexity results.

Table 1.2: Selected known complexity results for solving the WBP with T sample distributions and m supports. (C represents the largest entries of the cost matrix, while R_1 denotes the distance between the initial point and the solution set, and R_2 in the complexity bound of the sGS-ADMM comes from the additional proximal term.)

Algorithm	Time Complexity Result
IBP (Benamou et al., 2015; Kroshnin et al., 2019)	$\tilde{O}(C^2 T m^2 / \varepsilon^2)$
PDAGD (Kroshnin et al., 2019)	$\tilde{O}(C T m^{5/2} / \varepsilon)$
FastIBP (Lin et al., 2020)	$\tilde{O}(C^{4/3} (T m^{7/3} / \varepsilon^{4/3}))$
Dual extrapolation with area-convexity (Dvinskikh and Tiapkin, 2021)	$\tilde{O}(C T m^2 / \varepsilon)$
Mirror-Prox (Dvinskikh and Tiapkin, 2021)	$\tilde{O}(C T m^{5/2} / \varepsilon)$
Accelerated alternating minimization (Guminov et al., 2021)	$\tilde{O}(C T m^{5/2} / \varepsilon)$
Accelerated Bregman primal-dual method (Chambolle and Contreras, 2022)	$\tilde{O}(C T m^{5/2} / \varepsilon)$
sGS-ADMM (for dual WBP) (Yang et al., 2021; Cui et al., 2016)	$O((R_1 + R_2) T m^2 / \varepsilon^2)$
HPR (This thesis)	$O(R_1 T m^2 / \varepsilon)$

1.2 Contributions

To address the computational challenges in large-scale OT-related problems in the discrete setting, this thesis develops scalable and accelerated ADMM-type algorithms for efficiently solving linearly constrained convex optimization problems (COPs)—a broad class that includes linear programs as special cases, and thus subsumes many fundamental discrete OT tasks such as computing the KW distance and the Wasserstein barycenter with pre-specified finite supports. Building on the strong performance of these algorithms on OT-related problems, we further extend the methodology to general large-scale LP problems. The main contributions are summarized as follows:

- First, we propose the Halpern–Peaceman–Rachford (HPR) algorithm without proximal terms for solving COPs, by applying Halpern iteration to the Peaceman–Rachford splitting method. The algorithm achieves a non-ergodic iteration complexity of $O(1/\varepsilon)$ for computing an ε -approximate solution in terms of the KKT residual and the objective error. When applied to the WBP with pre-specified finite supports involving T probability distributions, each supported on m points in $\mathbb{R}^d$, we derive closed-form solutions for the subproblems in the HPR iteration and attain an overall computational complexity of $O(Tm^2/\varepsilon)$ flops—improving upon the best-known bound $\tilde{O}(Tm^2/\varepsilon)$ by eliminating extraneous logarithmic factors. Extensive numerical experiments on both synthetic and real datasets further demonstrate the superior performance of the HPR algorithm for solving large-scale WBPs.
- Second, we develop an accelerated preconditioned ADMM (pADMM) method with semi-proximal terms for solving COPs, where the HPR algorithm emerges as a special case when the proximal terms are removed. To accelerate pADMM,

we reformulate it as a proximal point algorithm (PPA) equipped with a positive semidefinite, possibly degenerate preconditioner, due to the lack of strong convexity in the proximal terms. By integrating Halpern iteration and the fast Krasnosel’skiĭ–Mann iteration into the resulting degenerate PPA (dPPA), we establish both asymptotic $o(1/k)$ and non-asymptotic $O(1/k)$ iteration complexity. Building on this foundation, we develop an accelerated pADMM algorithm that achieves these rates for both the KKT residual and the objective error. We further apply this framework to develop HOT, a Halpern accelerating algorithm for computing the KW distance. Rather than solving the original OT problem directly, HOT solves an equivalent reduced formulation that leverages the problem structure to reduce computational complexity. For discrete OT problems with m support points in $\mathbb{R}^2$ under an L_2^2 -ground cost, we derive closed-form solutions for the subproblems in HOT, resulting in an overall computational complexity of $O(m^{1.5}/\varepsilon)$ flops—surpassing the previous best-known bound of $\tilde{O}(m^2/\varepsilon)$. To address the key limitation that reduced OT models do not directly yield the OT plan, we propose an efficient recovery procedure that reconstructs the transport plan for the original problem from the solution of the reduced formulation. We implement HOT in PyTorch, and extensive numerical experiments demonstrate its superior performance compared to existing state-of-the-art algorithms for large-scale OT problems.

- Finally, motivated by the strong performance of the HPR method on OT-related problems, we extend the framework to general large-scale LP problems. We introduce HPR-LP, an implementation of the HPR method with semi-proximal terms for solving large-scale general LP problems. Based on the $O(1/k)$ complexity results of the HPR method, we design an adaptive strategy of restart and penalty parameter update to improve the efficiency and robust-

ness of the HPR method. We conduct extensive numerical experiments on different LP benchmark datasets using an NVIDIA A100-SXM4-80GB GPU under different stopping tolerances. The Julia implementation of HPR-LP 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} .

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 presents the necessary preliminaries for the developments in subsequent chapters. We review the concepts and recent advancements in dPPA, fixed-point methods, and pADMM, which serve as the theoretical foundation for our proposed algorithms.

Chapter 3 first introduces the HPR method for solving monotone inclusion problems. Building on this framework, we propose the HPR algorithm without proximal terms for solving COPs. We then apply the HPR algorithm to the WBP with pre-specified finite supports, where we derive closed-form solutions for the subproblems and design an efficient implementation. Extensive numerical experiments demonstrate the superior performance of the HPR method on large-scale WBP instances.

Chapter 4 develops an accelerated dPPA framework for monotone inclusion problems and establishes the equivalence between pADMM and dPPA. Leveraging this connection, we design an accelerated pADMM algorithm with provable convergence rates. We further apply this framework to develop the HOT algorithm—a Halpern accelerating method for solving a reduced formulation of the discrete OT problem. We also propose a transport plan recovery procedure that overcomes the limitations of reduced models. Extensive numerical results show that HOT outper-

forms existing state-of-the-art algorithms on large-scale discrete OT problems.

Chapter 5 presents HPR-LP, a practical implementation of the HPR method tailored for solving large-scale general LP problems. The method incorporates adaptive restart strategies and penalty parameter updates. We evaluate its performance on benchmark LP datasets, demonstrating improvements in speed and robustness compared to leading solvers such as PDLP.

Chapter 6 concludes the thesis and outlines potential directions for future research.

Chapter 2

Preliminaries

2.1 Notation

1. Let $\mathbb{U}$, $\mathbb{W}$, $\mathbb{X}$, $\mathbb{Y}$, and $\mathbb{Z}$ be finite-dimensional real Hilbert spaces, each equipped with an inner product $\langle \cdot, \cdot \rangle$ and the corresponding norm $\| \cdot \|$.
2. For any convex function $f : \mathbb{X} \rightarrow (-\infty, +\infty]$, we define its effective domain as $\text{dom}(f) := \{x \in \mathbb{X} \mid f(x) < \infty\}$. Its convex conjugate is given by $f^*(y) := \sup_{x \in \mathbb{X}} \{\langle y, x \rangle - f(x)\}$ for $y \in \mathbb{X}$. The subdifferential of f at x is $\partial f(x) := \{v \in \mathbb{X} \mid f(z) \geq f(x) + \langle v, z - x \rangle, \forall z \in \mathbb{X}\}$. In addition, the Moreau–Yosida proximal mapping of f is denoted by $\text{Prox}_f(x) := \arg \min_{z \in \mathbb{X}} \{f(z) + \frac{1}{2}\|z - x\|^2\}$ for $x \in \mathbb{X}$. Moreover, there exists a self-adjoint and positive semidefinite operator Σ_f such that for all $x, \hat{x} \in \text{dom}(f)$ and $\hat{\phi} \in \partial f(\hat{x})$,

$$f(x) \geq f(\hat{x}) + \langle \hat{\phi}, x - \hat{x} \rangle + \frac{1}{2}\|x - \hat{x}\|_{\Sigma_f}^2.$$

3. For a linear operator $A : \mathbb{X} \rightarrow \mathbb{Y}$, we denote its adjoint by A^* , its spectral norm by $\|A\|$, and the largest eigenvalue of AA^* by $\lambda_1(AA^*)$. For a matrix $B \in \mathbb{R}^{m \times n}$, we denote its transpose by $B^\top \in \mathbb{R}^{n \times m}$, which coincides with its adjoint B^* . The Frobenius norm of B is defined as $\|B\|_F := \sqrt{\text{trace}(BB^\top)}$, and the element-wise infinity norm as $\|B\|_\infty := \max_{1 \leq i \leq m, 1 \leq j \leq n} |B_{ij}|$, where

$\text{trace}(BB^\top)$ denotes the trace of the matrix $BB^\top$.

4. Let $\mathbf{1}_m$ (resp. $\mathbf{0}_m$) denote the m -dimensional vector of all ones (resp. all zeros).

The symbol $\otimes$ denotes the Kronecker product. For a matrix $X \in \mathbb{R}^{m \times n}$, we denote its vectorization by $\text{vec}(X) \in \mathbb{R}^{mn}$. For a collection of matrices $\{A_1, \dots, A_m\}$, we use $\text{diag}(A_1, \dots, A_m)$ to denote the block-diagonal matrix with diagonal blocks A_i .

5. For any self-adjoint, positive semidefinite linear operator $\mathcal{M} : \mathbb{X} \rightarrow \mathbb{X}$, we define the $\mathcal{M}$ -inner product and the associated seminorm by

$$\langle u, v \rangle_{\mathcal{M}} := \langle u, \mathcal{M}v \rangle, \quad \|x\|_{\mathcal{M}} := \sqrt{\langle x, x \rangle_{\mathcal{M}}} = \sqrt{\langle x, \mathcal{M}x \rangle}, \quad \forall u, v, x \in \mathbb{X}.$$

Moreover, the operator $\sqrt{\mathcal{M}}$ denotes the self-adjoint, positive semidefinite square root of $\mathcal{M}$, satisfying $\sqrt{\mathcal{M}}\sqrt{\mathcal{M}} = \mathcal{M}$.

6. For a nonexpansive operator $\mathcal{F} : \mathbb{X} \rightarrow \mathbb{X}$, the set of its fixed-points is denoted by $\text{Fix}\mathcal{F} := \{x \in \mathbb{X} \mid \mathcal{F}(x) = x\}$.

7. For any maximal monotone operator $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$, its resolvent is defined as $\mathcal{J}_{\mathcal{T}} := (\mathcal{I} + \mathcal{T})^{-1}$. The set of zeros of $\mathcal{T}$ is denoted by $\text{Zer}(\mathcal{T}) := \mathcal{T}^{-1}(0) = \{w \in \mathbb{W} \mid 0 \in \mathcal{T}w\}$.

8. Given a closed convex set $C \subseteq \mathbb{X}$, the Euclidean projection of x onto C is defined as $\Pi_C(x) := \arg \min_{z \in C} \|x - z\|$. For any $x \in C$, we use $\mathcal{N}_C(x)$ to denote the normal cone of C at x .

2.2 Degenerate Proximal Point Algorithm

A set-valued operator $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ is said to be a monotone operator if it satisfies the following inequality

$$\langle v - v', w - w' \rangle \geq 0 \quad \text{whenever} \quad v \in \mathcal{T}w, v' \in \mathcal{T}w'.$$

It is said to be maximal monotone if, in addition, the graph

$$\text{gph}(\mathcal{T}) = \{(w, v) \in \mathbb{W} \times \mathbb{W} \mid v \in \mathcal{T}w\}$$

is not properly contained in the graph of any other monotone operator $\mathcal{T}' : \mathbb{W} \rightarrow 2^{\mathbb{W}}$.

Consider the following monotone inclusion problem:

$$\text{find } w \in \mathbb{W} \text{ such that } 0 \in \mathcal{T}w, \quad (2.1)$$

where $\mathcal{T}$ is a maximal monotone operator from $\mathbb{W}$ into itself. If one introduces a preconditioner, namely, a linear, bounded, self-adjoint, and positive semidefinite operator $\mathcal{M} : \mathbb{W} \rightarrow \mathbb{W}$, then the proximal point algorithm (PPA) with the preconditioner $\mathcal{M}$ (Bredies et al., 2022) for solving (2.1) can be expressed as follows:

$$w^0 \in \mathbb{W}, \quad w^{k+1} = \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k := (\mathcal{M} + \mathcal{T})^{-1} \mathcal{M} w^k. \quad (2.2)$$

When $\mathcal{M} = \mathcal{I}$, scheme (2.2) reduces to the classical PPA (Rockafellar, 1976b,a). Proper choices of $\mathcal{M}$ will allow for efficient evaluations of $\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$. To ensure the well-definedness of (2.2), we introduce the concept of an admissible preconditioner:

Definition 2.1 (admissible preconditioner (Bredies et al., 2022)). *An admissible preconditioner for the operator $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ is a linear, bounded, self-adjoint, and positive semidefinite operator $\mathcal{M} : \mathbb{W} \rightarrow \mathbb{W}$ such that*

$$\mathcal{J}_{\mathcal{T}}^{\mathcal{M}} = (\mathcal{M} + \mathcal{T})^{-1} \mathcal{M} \quad (2.3)$$

is single-valued and has full domain.

Drawing from classical results in functional operator theory, Bredies et al. (Bredies et al., 2022) provided the following fundamental decomposition result to characterize linear, bounded, self-adjoint, and positive semidefinite operators.

Proposition 2.1 (Proposition 2.3 in (Bredies et al., 2022)). *Let $\mathcal{M} : \mathbb{W} \rightarrow \mathbb{W}$ be a linear, bounded, self-adjoint, and positive semidefinite operator. Then, there exists a bounded and injective operator $\mathcal{C} : \mathbb{U} \rightarrow \mathbb{W}$ such that $\mathcal{M} = \mathcal{C}\mathcal{C}^*$.*

Let $\mathcal{M}$ be an admissible preconditioner for the maximal monotone operator $\mathcal{T}$. The dPPA for solving the inclusion problem (2.1) is expressed as follows:

$$w^0 \in \mathbb{W}, \quad w^{k+1} = (1 - \rho_k)w^k + \rho_k \bar{w}^k, \quad \bar{w}^k = \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k = (\mathcal{M} + \mathcal{T})^{-1} \mathcal{M} w^k, \quad (2.4)$$

where $\{\rho_k\}$ is a sequence in $[0, 2]$. Here, $\mathcal{M}$ is only required to be positive semidefinite, which is the reason we refer to it as a dPPA. For notational convenience, define the following mappings:

$$\mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} := \mathcal{I} - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} \quad \text{and} \quad \mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} := (1 - \rho)\mathcal{I} + \rho\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}, \quad \rho \in [0, 2], \quad (2.5)$$

where $\mathcal{I}$ is the identity operator on $\mathbb{W}$. Clearly, if $0 \in \mathcal{T}w$, we have that $\mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w = w$ and $\mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w = 0$. Similar to (Rockafellar, 1976b, Proposition 1), we summarize some properties of $\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$, $\mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}}$, and $\mathcal{Q}_{\mathcal{T}}^{\mathcal{M}}$ in the following proposition:

Proposition 2.2. *The following things hold:*

(a) $w = \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w + \mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w$ and $\mathcal{M}\mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w \in \mathcal{T}(\mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w)$ for all $w \in \mathbb{W}$;

(b) $\langle \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w', \mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w - \mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w' \rangle_{\mathcal{M}} \geq 0$ for all $w, w' \in \mathbb{W}$;

(c) $\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$ is $\mathcal{M}$ -firmly nonexpansive, i.e.,

$$\|\mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w'\|_{\mathcal{M}}^2 + \|\mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w - \mathcal{Q}_{\mathcal{T}}^{\mathcal{M}} w'\|_{\mathcal{M}}^2 \leq \|w - w'\|_{\mathcal{M}}^2 \quad \text{for all } w, w' \in \mathbb{W};$$

(d) $\mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}}$ is $\mathcal{M}$ -nonexpansive for $\rho \in (0, 2]$, i.e.,

$$\|\mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} w - \mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} w'\|_{\mathcal{M}} \leq \|w - w'\|_{\mathcal{M}} \quad \text{for all } w, w' \in \mathbb{W}.$$

Proof. The proofs of parts (a)-(c) can be obtained through straightforward calculations in a way similar to the proof provided in (Rockafellar, 1976b, Proposition 1). Part (d) follows from part (c), which can be verified by referring to (Bauschke and Combettes, 2017, Exercise 4.13), with $\|\cdot\|$ being replaced by $\|\cdot\|_{\mathcal{M}}$. We omit the details here. $\square$

Note that when $\mathcal{M}$ is the identity operator, $\mathcal{M}$ -firmly nonexpansiveness and $\mathcal{M}$ -nonexpansiveness reduce to the firmly nonexpansiveness and nonexpansiveness, respectively. Furthermore, the global convergence of the dPPA in (2.4) can be established under the assumption that $(\mathcal{M} + \mathcal{T})^{-1}$ is L -Lipschitz; that is, there exists a constant $L \geq 0$ such that for all v_1, v_2 in the domain of $(\mathcal{M} + \mathcal{T})^{-1}$,

$$\|(\mathcal{M} + \mathcal{T})^{-1}v_1 - (\mathcal{M} + \mathcal{T})^{-1}v_2\| \leq L\|v_1 - v_2\|.$$

Bredies et al. (Bredies et al., 2022) showed that this Lipschitz assumption holds for many splitting algorithms, including the Douglas–Rachford splitting method (Lions and Mercier, 1979) and the Chambolle–Pock scheme (Chambolle and Pock, 2011), under mild conditions. Additional examples can be found in Section 3 of (Bredies et al., 2022).

Theorem 2.1 (Corollary 2.10 in (Bredies et al., 2022)). *Let $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ with $\mathcal{T}^{-1}(0) \neq \emptyset$ be a maximal monotone operator and let $\mathcal{M}$ be an admissible preconditioner such that $(\mathcal{M} + \mathcal{T})^{-1}$ is L -Lipschitz. Let $\{w^k\}$ be any sequence generated by the dPPA in (2.4). If $0 < \inf_k \rho_k \leq \sup_k \rho_k < 2$, then $\{w^k\}$ converges to a point in $\mathcal{T}^{-1}(0)$.*

Furthermore, Bredies et al. (Bredies et al., 2022) introduced a reduced preconditioned PPA for the parallel composition of $\mathcal{T}$ by $\mathcal{C}^*$, which can help us explore the acceleration of the dPPA in (2.4). We summarize some key results related to parallel composition in the following proposition.

Proposition 2.3. *Let $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ be a maximal monotone operator and let $\mathcal{M}$ be an admissible preconditioner. Suppose that $\mathcal{M} = \mathcal{C}\mathcal{C}^*$ is a decomposition of $\mathcal{M}$ according to Proposition 2.1 with $\mathcal{C} : \mathbb{U} \rightarrow \mathbb{W}$. The parallel composition $\mathcal{C}^* \triangleright \mathcal{T} := (\mathcal{C}^* \mathcal{T}^{-1} \mathcal{C})^{-1}$ is a maximal monotone operator. Furthermore, the resolvent of $\mathcal{C}^* \triangleright \mathcal{T}$ denoted by $\mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}} := (\mathcal{I} + \mathcal{C}^* \triangleright \mathcal{T})^{-1}$, has the following identity*

$$\mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}} = \mathcal{C}^* (\mathcal{M} + \mathcal{T})^{-1} \mathcal{C}. \quad (2.6)$$

In particular, $\mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}} : \mathbb{U} \rightarrow \mathbb{U}$ is everywhere well-defined and firmly nonexpansive. Moreover, for any $\rho \in (0, 2]$, $\mathcal{F}_{\mathcal{C}^* \triangleright \mathcal{T}, \rho} = (1 - \rho)\mathcal{I} + \rho\mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}}$ is nonexpansive and

$$\mathcal{C}^* \mathcal{T}^{-1}(0) = \mathcal{C}^* \text{Fix } \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} = \text{Fix } \mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}} = \text{Fix } \mathcal{F}_{\mathcal{C}^* \triangleright \mathcal{T}, \rho},$$

where we denote the set of fixed-points of an operator $\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$ by $\text{Fix } \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$.

Proof. See Lemma 2.12 and Theorem 2.13 in (Bredies et al., 2022) for the maximal monotonicity of $\mathcal{C}^* \triangleright \mathcal{T}$, the identity (2.6), and the firm nonexpansiveness of $\mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}}$. Moreover, it follows from (Bauschke and Combettes, 2017, Exercise 4.13) that for any $\rho \in (0, 2]$, $\mathcal{F}_{\mathcal{C}^* \triangleright \mathcal{T}, \rho}$ is nonexpansive. Finally, from the proof of Theorem 2.14 in (Bredies et al., 2022), we obtain $\mathcal{C}^* \text{Fix } \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} = \text{Fix } \mathcal{J}_{\mathcal{C}^* \triangleright \mathcal{T}}$. Combining this with the fact $\mathcal{T}^{-1}(0) = \text{Fix } \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$, we conclude the proof. $\square$

2.3 Fixed-point Iteration Methods

In this section, we review some fixed-point iteration methods, which can be used to accelerate pADMM. Consider the following fixed-point problem:

$$\text{Find } \eta \in \mathbb{X} \text{ such that } \eta = \mathcal{F}\eta, \quad (2.7)$$

where $\mathcal{F} : \mathbb{X} \rightarrow \mathbb{X}$ is a nonexpansive operator. A natural approach for finding a fixed-point of $\mathcal{F}$ is the Banach–Picard iteration:

$$\eta^{k+1} := \mathcal{F}\eta^k \quad \forall k \geq 0, \quad (2.8)$$

where $\eta^0 \in \mathbb{X}$ is an initial point. However, for a general nonexpansive operator $\mathcal{F}$, the Banach–Picard iteration may fail to converge.

In contrast, the Halpern iteration (Halpern, 1967) ensures convergence to the best approximation of η^0 from the set of fixed-points $\text{Fix } \mathcal{F}$, provided that $\text{Fix } \mathcal{F} \neq \emptyset$. Specifically, the Halpern iteration for solving the fixed-point problem (2.7) follows the iterative scheme:

$$\eta^{k+1} := \lambda_k \eta^0 + (1 - \lambda_k) \mathcal{F} \eta^k \quad \forall k \geq 0, \quad (2.9)$$

where $\lambda_k \in [0, 1]$ is a sequence of parameters. The following theorem provides the convergence guarantees for the Halpern iteration.

Theorem 2.2 (Theorem 2 in (Wittmann, 1992)). *Let D be a nonempty closed convex subset of $\mathbb{X}$, and let $\mathcal{F} : D \rightarrow D$ be a nonexpansive operator such that $\text{Fix } \mathcal{F} \neq \emptyset$. Let $\{\lambda_k\}_{k=0}^\infty$ be a sequence in $[0, 1]$ such that the following hold:*

$$\lambda_k \rightarrow 0, \quad \sum_{k=0}^\infty \lambda_k = +\infty, \quad \sum_{k=0}^\infty |\lambda_{k+1} - \lambda_k| < +\infty.$$

Let $\eta^0 \in D$ and set

$$\eta^{k+1} := \lambda_k \eta^0 + (1 - \lambda_k) \mathcal{F} \eta^k \quad \forall k \geq 0.$$

Then $\eta^k \rightarrow \Pi_{\text{Fix } \mathcal{F}}(\eta^0)$.

Recently, Lieder (2021) showed that choosing $\lambda_k = 1/(k+2)$ for any $k \geq 0$ leads to an optimal iteration complexity for the residual in the Halpern iteration.

Theorem 2.3 (Theorem 2.1 in (Lieder, 2021)). *Let $\mathcal{F} : \mathbb{X} \rightarrow \mathbb{X}$ be a nonexpansive operator such that $\text{Fix } \mathcal{F} \neq \emptyset$. Then the sequence generated by the Halpern iteration (2.9) with $\lambda_k = 1/(k+2)$ for any $k \geq 0$ satisfies*

$$\|\eta^k - \mathcal{F} \eta^k\| \leq \frac{2\|\eta^0 - \eta^*\|}{k+1} \quad \forall k \geq 0, \quad \text{for any } \eta^* \in \text{Fix } \mathcal{F}. \quad (2.10)$$

Moreover, this bound is tight.

It is worth noting that Contreras and Cominetti (2023) demonstrated that the best possible convergence rate for general Mann iterations in normed spaces, including the Halpern iteration, is lower bounded by $O(1/k)$. In contrast, inspired by the second-order dynamical system with a vanishing damping term proposed in (Boţ et al., 2025) for solving monotone equations, Boţ and Nguyen (2023) introduced the fast Krasnosel’skiĭ–Mann (KM) iteration with asymptotic $o(1/k)$ convergence rates for finding a fixed-point of the nonexpansive operator, which appears to offer better convergence rates than $O(1/k)$ in certain applications. Specifically, given $\alpha > 2$, η_0 and $\eta_1 \in \mathbb{X}$, the fast KM method with stepsize 1 iterates as follows:

$$\eta^{k+1} = \eta^k + \frac{\alpha}{2(k+\alpha)}(\mathcal{F}\eta^k - \eta^k) + \frac{k}{k+\alpha}(\mathcal{F}\eta^k - \mathcal{F}\eta^{k-1}) \quad \forall k \geq 1. \quad (2.11)$$

The convergence result of the fast KM method is established in the following theorem.

Theorem 2.4 (Theorem 3.4 in (Boţ and Nguyen, 2023)). *Let $\mathcal{F} : \mathbb{X} \rightarrow \mathbb{X}$ be a nonexpansive operator such that $\text{Fix } \mathcal{F} \neq \emptyset$. Then the sequence $\{\eta^k\}$ generated by the fast KM method in (2.11) converges to an element in $\text{Fix } \mathcal{F}$.*

Furthermore, the iteration complexity of the fast KM method is established in the following result.

Theorem 2.5 (Theorem 3.5 in (Boţ and Nguyen, 2023)). *Let $\mathcal{F} : \mathbb{X} \rightarrow \mathbb{X}$ be a nonexpansive operator such that $\text{Fix } \mathcal{F} \neq \emptyset$. Then the sequence $\{\eta^k\}$ generated by the fast KM method in (2.11) satisfies:*

$$\|\eta^{k+1} - \eta^k\| = o\left(\frac{1}{k}\right) \quad \text{and} \quad \|\eta^k - \mathcal{F}\eta^k\| = o\left(\frac{1}{k}\right) \quad \text{as } k \rightarrow +\infty.$$

2.4 Preconditioned (Semi-proximal) ADMM

Consider the following convex optimization problem:

$$\begin{aligned} \min_{y \in \mathbb{Y}, z \in \mathbb{Z}} \quad & f_1(y) + f_2(z) \\ \text{subject to} \quad & B_1 y + B_2 z = c, \end{aligned} \quad (2.12)$$

where $f_1 : \mathbb{Y} \rightarrow (-\infty, +\infty]$ and $f_2 : \mathbb{Z} \rightarrow (-\infty, +\infty]$ are two proper closed convex functions, $B_1 : \mathbb{Y} \rightarrow \mathbb{X}$ and $B_2 : \mathbb{Z} \rightarrow \mathbb{X}$ 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 (2.12) is defined by, for any $(y, z, x) \in \mathbb{Y} \times \mathbb{Z} \times \mathbb{X}$,

$$L_\sigma(y, z; x) := f_1(y) + f_2(z) + \langle x, B_1 y + B_2 z - c \rangle + \frac{\sigma}{2} \|B_1 y + B_2 z - c\|^2.$$

The dual of problem (2.12) is given by

$$\max_{x \in \mathbb{X}} \{-f_1^*(-B_1^* x) - f_2^*(-B_2^* x) - \langle c, x \rangle\}, \quad (2.13)$$

where $B_1^* : \mathbb{X} \rightarrow \mathbb{Y}$ and $B_2^* : \mathbb{X} \rightarrow \mathbb{Z}$ are the adjoint of B_1 and B_2 , respectively. For ease of notation, let $w := (y, z, x)$ and $\mathbb{W} := \mathbb{Y} \times \mathbb{Z} \times \mathbb{X}$. For any self-adjoint positive semidefinite linear operator $\mathcal{M} : \mathbb{X} \rightarrow \mathbb{X}$, denote $\|x\|_{\mathcal{M}} := \sqrt{\langle x, \mathcal{M} x \rangle}$. In extending the framework of ADMM with semi-proximal terms and larger dual step lengths studied in (Fazel et al., 2013) to cover the case of generalized ADMM in the sense of Eckstein and Bertsekas (1992), Xiao et al. (2018) considered the pADMM in Algorithm 1 for solving problem (2.12):

Algorithm 1 A pADMM for solving convex optimization problem (2.12)

- 1: **Input:** Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be self-adjoint positive semidefinite operators on $\mathbb{Y}$ and $\mathbb{Z}$, respectively. Choose $w^0 = (y^0, z^0, x^0) \in \text{dom}(f_1) \times \text{dom}(f_2) \times \mathbb{X}$. Set $\sigma > 0$ and $\rho_k \in (0, 2]$ for all $k \geq 0$.
 - 2: **for** $k = 0, 1, \dots$ **do**
 - 3: Step 1. $\bar{z}^k = \arg \min_{z \in \mathbb{Z}} \left\{ L_\sigma(y^k, z; x^k) + \frac{1}{2} \|z - z^k\|_{\mathcal{T}_2}^2 \right\}$
 - 4: Step 2. $\bar{x}^k = x^k + \sigma(B_1 y^k + B_2 \bar{z}^k - c)$
 - 5: Step 3. $\bar{y}^k = \arg \min_{y \in \mathbb{Y}} \left\{ L_\sigma(y, \bar{z}^k; \bar{x}^k) + \frac{1}{2} \|y - y^k\|_{\mathcal{T}_1}^2 \right\}$
 - 6: Step 4. $w^{k+1} = (1 - \rho_k)w^k + \rho_k \bar{w}^k$
-

Since the two linear operators $\mathcal{T}_1$ and $\mathcal{T}_2$ in Algorithm 1 are only assumed to be positive semidefinite, the pADMM framework includes ADMM (Gabay and Mercier,

1976; Glowinski and Marroco, 1975), proximal ADMM (Eckstein, 1994), and semi-proximal ADMM (Fazel et al., 2013) as special cases. Specifically, when $\mathcal{T}_1 = 0$, $\mathcal{T}_2 = 0$, and $\rho_k \equiv 1$, pADMM corresponds to the ADMM (Gabay and Mercier, 1976; Glowinski and Marroco, 1975) with a unit dual step-length. Additionally, if both $\mathcal{T}_1$ and $\mathcal{T}_2$ are positive definite, then the above pADMM scheme with $\rho_k \equiv 1$ reduces to the proximal ADMM proposed by Eckstein (1994) with possible changes of inner products. Moreover, when $\rho_k \equiv 1$, pADMM corresponds to a special case studied in (Fazel et al., 2013) under the setting of semi-proximal ADMM. On the other hand, it is worth highlighting that pADMM is capable of handling multi-block convex optimization problems by introducing the symmetric Gauss-Seidel (sGS) operator (Li et al., 2016, 2019) as a linear operator in the proximal term, which has proven to be effective in solving large-scale optimization problems (Li et al., 2016, 2018; Liang et al., 2022; Xiao et al., 2018). Therefore, given the broad applicability and effectiveness of the pADMM framework, investigating the acceleration of pADMM is of significant interest.

Before discussing the acceleration techniques, we mention several developments in the convergence rate analysis of pADMM that are relevant to this thesis: Monteiro and Svaiter (2013) first established an ergodic $O(1/k)$ convergence rate for the ADMM with a unit dual stepsize in terms of the Karush-Kuhn-Tucker (KKT) type residual; Davis and Yin (2016) provided a non-ergodic iteration complexity bound of $o(1/\sqrt{k})$ for the ADMM with a unit dual stepsize, focusing on feasibility violations and the objective error; and Cui et al. (2016) demonstrated that a majorized ADMM, including the classical ADMM, exhibits a non-ergodic $O(1/\sqrt{k})$ convergence rate concerning the KKT optimality condition.

To further enhance the efficiency of the pADMM, researchers have explored two main approaches for acceleration. One approach involves integrating Nesterov’s extrapolation (Nesterov, 1983; Beck and Teboulle, 2009) directly into the pADMM to

develop accelerated variants. For instance, when one of the objective functions is strongly convex, Xu (2017) proposed an accelerated linearized ADMM (LADMM) with an ergodic $O(1/k^2)$ convergence rate concerning feasibility violations and objective error. For more results on the strongly convex case, one may refer to (Goldstein et al., 2014; Tang and Toh, 2024) and the references therein. On the other hand, in the absence of a strong convexity assumption, which is a primary focus of this thesis, several accelerated ADMM versions with a convergence rate of $O(1/k)$ have been introduced. Specifically, assuming $f_2(\cdot)$ to be a smooth convex function with L_{f_2} -Lipschitz continuous gradient, Ouyang et al. (2015) proposed an accelerated variant of LADMM. In terms of function values and feasibility violations, the ergodic convergence rate of this method associated with the L_{f_2} part surpasses $O(1/k)$, while the ergodic rate for other parts remains $O(1/k)$ (see Tables 2 and 3 in (Ouyang et al., 2015)). Turning to the non-ergodic rates along this approach, two notable works are those of Li and Lin (2019) and Sabach and Teboulle (2022).

In (Li and Lin, 2019), Li and Lin modified the accelerated LADMM proposed in (Ouyang et al., 2015) to obtain a non-ergodic $O(1/(1 + k(1 - \tau)))$ convergence rate in terms of function values and feasibility violations with the dual step length τ to be restricted in $(0.5, 1)$. In their algorithm, the largest eigenvalues $\lambda_{\max}(\mathcal{T}_i^k)$ of $\mathcal{T}_i^k$ for $i = 1, 2$ vary with k and satisfy

$$\lambda_{\max}(\mathcal{T}_1^k) \sim O(1 + k(1 - \tau)), \quad \lambda_{\max}(\mathcal{T}_2^k) \sim O(1 + k(1 - \tau))$$

at the k -th iteration, as indicated in formulas (6a) and (6b) of (Li and Lin, 2019). This implies that the primal step length approaches zero as k tends to infinity. Furthermore, when the dual step length τ approaches one, the $O(1/(1 + k(1 - \tau)))$ complexity result will be lost (see Theorem 1 of (Li and Lin, 2019)). Different from (Li and Lin, 2019), Sabach and Teboulle (2022) introduced an accelerated variant of pADMM with a non-ergodic $O(1/k)$ convergence rate in terms of function values

and feasibility violations when $\mathcal{T}_2$ is positive definite. Note that the dual step length μ in Sabach and Teboulle's algorithm is required to satisfy

$$\mu \in (0, \delta], \quad \delta = 1 - \frac{\sigma \lambda_{\max}(B_2^* B_2)}{(\sigma \lambda_{\max}(B_2^* B_2) + \lambda_{\min}(\mathcal{T}_2))} < 1$$

according to Lemma 5.7 of (Sabach and Teboulle, 2022), where $\lambda_{\min}(\mathcal{T}_2)$ is the smallest eigenvalue of $\mathcal{T}_2$. When $\mathcal{T}_2$ tends to be degenerate, i.e., $\lambda_{\min}(\mathcal{T}_2)$ goes to zero, the dual step length μ approaches zero as δ tends to zero, which implies that the obtained complexity bounds blow up (see formulas (4.11) and (4.12) in Theorem 4.5 of (Sabach and Teboulle, 2022)). Thus, to the best of our knowledge, there is still a big gap in using this acceleration technique to handle the case where both $\mathcal{T}_1$ and $\mathcal{T}_2$ are positive semidefinite and/or with large step lengths.

Another approach for accelerating the pADMM is first to reformulate it as a fixed-point iterative method, if possible, and then to accelerate the pADMM by accelerating the fixed-point iterative method. For instance, Kim (2021) introduced an accelerated PPA (Rockafellar, 1976a,b) for the purpose of achieving a fast rate of $O(1/k)$. By reformulating the ADMM as the Douglas–Rachford (DR) splitting method, which is a special case of the PPA (Eckstein and Bertsekas, 1992), Kim (2021) obtained an accelerated ADMM that possesses a non-ergodic $O(1/k)$ convergence rate concerning the primal feasibility violations. Since Contreras and Cominetti (2023) established a close connection between Kim's accelerated PPA and Halpern's iteration (Halpern, 1967; Lieder, 2021; Sabach and Shtern, 2017), it is natural to employ the Halpern iteration to obtain an accelerated ADMM, owing to the simplicity of the Halpern iteration. Following this line, Tran-Dinh and Luo (2021) obtained an accelerated ADMM by accelerating the DR splitting method, which achieves a non-ergodic $O(1/k)$ convergence rate on the forward-backward residual operator associated with the dual problem (2.13). When $f_2(\cdot)$ is a strongly convex function, this result actually represents the $O(1/k)$ convergence rate in terms of primal feasibility violations.

Compared to the first approach of employing Nesterov's extrapolation to obtain an accelerated pADMM, the second approach mentioned above does not impose restrictive requirements on the step lengths. This inspires us to further explore accelerating the pADMM with semi-proximal terms by focusing on the second approach.

Chapter 3

A Halpern Peaceman–Rachford (HPR) Method for Solving Wasserstein Barycenter Problems

In this chapter, we first introduce the HPR method for solving monotone inclusion problems. Building on this framework, we propose an HPR algorithm without proximal terms for solving COPs. We then apply the HPR algorithm to the WBP, where we derive closed-form solutions for the subproblems and design an efficient implementation. Extensive numerical experiments demonstrate the superior performance of the HPR method on large-scale WBP instances.

3.1 An HPR Method for Convex Optimization Problems

Recall the linearly constrained COP (2.12) given in Chapter 2:

$$\begin{aligned} \min_{y \in \mathbb{Y}, z \in \mathbb{Z}} \quad & f_1(y) + f_2(z) \\ \text{subject to} \quad & B_1 y + B_2 z = c, \end{aligned} \tag{3.1}$$

where $f_1 : \mathbb{Y} \rightarrow (-\infty, +\infty]$ and $f_2 : \mathbb{Z} \rightarrow (-\infty, +\infty]$ are two proper closed convex functions, $B_1 : \mathbb{Y} \rightarrow \mathbb{X}$ and $B_2 : \mathbb{Z} \rightarrow \mathbb{X}$ 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 (3.1) is defined by, for any $(y, z, x) \in \mathbb{Y} \times \mathbb{Z} \times \mathbb{X}$,

$$L_\sigma(y, z; x) := f_1(y) + f_2(z) + \langle x, B_1 y + B_2 z - c \rangle + \frac{\sigma}{2} \|B_1 y + B_2 z - c\|^2.$$

The dual of problem (3.1) is given by

$$\max_{x \in \mathbb{X}} \{-f_1^*(-B_1^* x) - f_2^*(-B_2^* x) - \langle c, x \rangle\}, \quad (3.2)$$

where $B_1^* : \mathbb{X} \rightarrow \mathbb{Y}$ and $B_2^* : \mathbb{X} \rightarrow \mathbb{Z}$ are the adjoint of B_1 and B_2 , respectively. Since f_1 and f_2 are proper closed convex functions, there exist two self-adjoint and positive semidefinite operators Σ_{f_1} and Σ_{f_2} such that for all $y, \hat{y} \in \text{dom}(f_1)$, $\phi \in \partial f_1(y)$, and $\hat{\phi} \in \partial f_1(\hat{y})$,

$$f_1(y) \geq f_1(\hat{y}) + \langle \hat{\phi}, y - \hat{y} \rangle + \frac{1}{2} \|y - \hat{y}\|_{\Sigma_{f_1}}^2 \text{ and } \langle \phi - \hat{\phi}, y - \hat{y} \rangle \geq \|y - \hat{y}\|_{\Sigma_{f_1}}^2,$$

and for all $z, \hat{z} \in \text{dom}(f_2)$, $\varphi \in \partial f_2(z)$, and $\hat{\varphi} \in \partial f_2(\hat{z})$

$$f_2(z) \geq f_2(\hat{z}) + \langle \hat{\varphi}, z - \hat{z} \rangle + \frac{1}{2} \|z - \hat{z}\|_{\Sigma_{f_2}}^2 \text{ and } \langle \varphi - \hat{\varphi}, z - \hat{z} \rangle \geq \|z - \hat{z}\|_{\Sigma_{f_2}}^2.$$

In this section, we make the following assumptions:

Assumption 1. *For $i = 1, 2$, the following conditions hold:*

$$(A1.1) \text{ ri}(\text{dom } f_i^*) \cap \text{Range}(B_i^*) \neq \emptyset;$$

$$(A1.2) \text{ ri}(\text{dom}(f_1^* \circ (-B_1^*))) \cap \text{ri}(\text{dom}(f_2^* \circ (-B_2^*))) \neq \emptyset;$$

$$(A1.3) \text{ The solution set of the optimization problem (3.2) is nonempty.}$$

Assumption 2. *Both $\Sigma_{f_1} + B_1^* B_1$ and $\Sigma_{f_2} + B_2^* B_2$ are positive definite.*

Under Assumption 1, $(y^*, z^*) \in \mathbb{Y} \times \mathbb{Z}$ is a solution to the COP (3.1) and $x^* \in \mathbb{X}$ is a solution to the dual of COP (3.2) if and only if the following KKT conditions are satisfied:

$$B_1 y^* + B_2 z^* = c, \quad -B_2^* x^* \in \partial f_2(z^*), \quad -B_1^* x^* \in \partial f_1(y^*). \quad (3.3)$$

Furthermore, under Assumption 1, the dual of COP (3.2) is a special case of the following monotone inclusion problem:

$$0 \in (\mathcal{G}_1 + \mathcal{G}_2)x, \quad x \in \mathbb{X}, \quad (3.4)$$

where $\mathcal{G}_1 : \mathbb{X} \rightarrow 2^{\mathbb{X}}$, $\mathcal{G}_2 : \mathbb{X} \rightarrow 2^{\mathbb{X}}$, and $\mathcal{G}_1 + \mathcal{G}_2$ are all maximal monotone operators. The relaxed Peaceman–Rachford (PR) splitting method with the under-relaxation factor $\rho \in (0, 1]$ (Lions and Mercier, 1979) for solving problem (3.4) is given by

$$\eta^{k+1} = \mathcal{F}_{\sigma, \rho}^{\text{PR}}(\eta^k) := \mathcal{I} + 2\rho(\mathcal{J}_{\sigma\mathcal{G}_1} \circ (2\mathcal{J}_{\sigma\mathcal{G}_2} - \mathcal{I}) - \mathcal{J}_{\sigma\mathcal{G}_2})(\eta^k), \quad k = 0, 1, \dots, \quad (3.5)$$

where $\eta^0 \in \mathbb{X}$ is an initial point. The case $\rho = 1/2$ corresponds to the widely studied Douglas–Rachford (DR) splitting method, while $\rho = 1$ is referred to as the PR splitting method. For simplicity, when $\rho = 1$, we denote $\mathcal{F}_{\sigma}^{\text{PR}} := \mathcal{F}_{\sigma, 1}^{\text{PR}}$. From (3.5), we obtain

$$\eta^{k+1} = \mathcal{F}_{\sigma}^{\text{PR}}(\eta^k) := (2\mathcal{J}_{\sigma\mathcal{G}_1} - \mathcal{I}) \circ (2\mathcal{J}_{\sigma\mathcal{G}_2} - \mathcal{I})(\eta^k) \quad \forall k \geq 0. \quad (3.6)$$

If $\eta^* \in \text{Fix } \mathcal{F}_{\sigma}^{\text{PR}}$, then $x^* = \mathcal{J}_{\sigma\mathcal{G}_2}(\eta^*)$ is a solution to (3.4) (Lions and Mercier, 1979).

On the one hand, since $\mathcal{F}_{\sigma}^{\text{PR}} : \mathbb{X} \rightarrow \mathbb{X}$ is only a nonexpansive operator, the PR splitting method may fail to converge for general monotone inclusion problems (3.4) (see, e.g., (Monteiro and Sim, 2018)). On the other hand, the DR splitting method achieves only an $o(1/\sqrt{k})$ iteration complexity in terms of the fixed-point residual (Davis and Yin, 2016). To address the convergence issues of the PR splitting method for general monotone inclusion problems (3.4) and to potentially achieve better iteration complexity compared to the DR splitting method, we propose the Halpern PR (HPR) algorithm, which applies the Halpern iteration (2.9) to the PR splitting method (3.6). The HPR algorithm is formally presented in Algorithm 2.

Algorithm 2 An HPR algorithm for solving (3.4)

```

1: Input:  $\eta^0 \in \mathbb{X}$ 
2: for  $k = 0, 1, \dots$  do
3:    $x^{k+1/2} = \mathcal{J}_{\sigma\mathcal{G}_2}(\eta^k)$ 
4:    $x^{k+1} = \mathcal{J}_{\sigma\mathcal{G}_1}(2x^{k+1/2} - \eta^k)$ 
5:    $v^{k+1} = 2x^{k+1} - (2x^{k+1/2} - \eta^k)$ 
6:    $\eta^{k+1} = \frac{1}{k+2}\eta^0 + \frac{k+1}{k+2}v^{k+1}$ 

```

The convergence result of the HPR algorithm for solving the monotone inclusion problem (3.4) with general operators $\mathcal{G}_1$ and $\mathcal{G}_2$ is summarized in Corollary 3.1.

Corollary 3.1. *Let $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_1 + \mathcal{G}_2$ be maximal monotone operators on $\mathbb{X}$ satisfying $\text{Zer}(\mathcal{G}_1 + \mathcal{G}_2) \neq \emptyset$. Let $\{(x^{k+1/2}, x^{k+1}, v^{k+1}, \eta^{k+1})\}$ be the sequence generated by Algorithm 2. Let $\eta^* = \Pi_{\text{Fix } \mathcal{F}_\sigma^{\text{PR}}}(\eta^0)$ and $x^* = \mathcal{J}_{\sigma\mathcal{G}_2}(\eta^*)$. Then we have*

$$\eta^{k+1} \rightarrow \eta^*, \quad v^{k+1} \rightarrow \eta^*, \quad x^{k+1} \rightarrow x^*, \quad \text{and } x^{k+1/2} \rightarrow x^*,$$

and x^* is a solution to the problem (3.4).

Proof. Since $\text{Fix } \mathcal{F}_\sigma^{\text{PR}}$ is a closed convex set (Bauschke and Combettes, 2017, Corollary 4.24), $\Pi_{\text{Fix } \mathcal{F}_\sigma^{\text{PR}}}(\eta^0)$ is unique. The convergence of the sequence $\{\eta^{k+1}\}$ comes from Theorem 2.2. Since for all $k \geq 0$,

$$\eta^{k+1} = \frac{1}{k+2}\eta^0 + \frac{k+1}{k+2}v^{k+1},$$

we can directly obtain $v^{k+1} \rightarrow \eta^*$. It follows from $x^{k+1} - x^{k+1/2} = 1/2(v^{k+1} - \eta^k) \forall k \geq 0$ in Algorithm 2 that

$$x^{k+1} - x^{k+1/2} \rightarrow 0. \tag{3.7}$$

Since $\mathcal{J}_{\sigma\mathcal{G}_2}$ is nonexpansive and $x^{k+1/2} = \mathcal{J}_{\sigma\mathcal{G}_2}(\eta^k)$, we directly obtain $\lim_{k \rightarrow \infty} x^{k+1} = \lim_{k \rightarrow \infty} x^{k+1/2} = \mathcal{J}_{\sigma\mathcal{G}_2}(\eta^*) = x^*$. This completes the proof. $\square$

The iteration complexity of the HPR algorithm follows directly from Theorem 2.3 on the Halpern iteration.

Corollary 3.2. *If $\text{Fix } \mathcal{F}_\sigma^{\text{PR}} \neq \emptyset$, then the sequence generated by Algorithm 2 with $\lambda_k = 1/(k+2)$ for any $k \geq 0$ satisfies*

$$\|\eta^k - \mathcal{F}_\sigma^{\text{PR}}(\eta^k)\| \leq \frac{2\|\eta^0 - \eta^*\|}{k+1} \quad \forall k \geq 0, \quad \text{for any } \eta^* \in \text{Fix } \mathcal{F}_\sigma^{\text{PR}}. \quad (3.8)$$

To apply the HPR method described in Algorithm 2 to solve the COP (3.1) and its dual problem (3.2), we define the operators

$$\mathcal{G}_1(x) := \partial(f_1^* \circ (-B_1^*))(x) + c \quad \text{and} \quad \mathcal{G}_2(x) := \partial(f_2^* \circ (-B_2^*))(x), \quad \forall x \in \mathbb{X}. \quad (3.9)$$

Subsequently, applying Algorithm 2 to the dual problem (3.2) yields Algorithm 3.

Algorithm 3 A preliminary HPR algorithm for solving COP (3.1)

- 1: **Input:** $y^0 \in \text{dom}(f_1)$, $x^0 \in \mathbb{X}$, and $\sigma > 0$
 - 2: **Initialization:** $\tilde{x}^0 = x^0$, $\eta^0 = \tilde{x}^0 + \sigma(B_1 y^0 - c)$
 - 3: **for** $k = 0, 1, \dots$ **do**
 - 4: Step 1. $z^{k+1} = \arg \min_{z \in \mathbb{Z}} \left\{ f_2(z) + \langle \eta^k, B_2 z \rangle + \frac{\sigma}{2} \|B_2 z\|^2 \right\}$
 - 5: Step 2. $x^{k+1/2} = \eta^k + \sigma B_2 z^{k+1}$
 - 6: Step 3. $y^{k+1} = \arg \min_{y \in \mathbb{Y}} \left\{ f_1(y) + \langle \eta^k + 2\sigma B_2 z^{k+1}, B_1 y - c \rangle + \frac{\sigma}{2} \|B_1 y - c\|^2 \right\}$
 - 7: Step 4. $x^{k+1} = \eta^k + \sigma(B_1 y^{k+1} - c) + 2\sigma B_2 z^{k+1}$
 - 8: Step 5. $v^{k+1} = \eta^k + 2\sigma(B_1 y^{k+1} + B_2 z^{k+1} - c)$
 - 9: Step 6. $\eta^{k+1} = \frac{1}{k+2} \eta^0 + \frac{k+1}{k+2} v^{k+1}$
-

In Algorithm 3, if we define

$$\tilde{x}^{k+1} := \eta^{k+1} - \sigma(B_1 y^{k+1} - c) \quad \forall k \geq 0, \quad (3.10)$$

we can discard the sequences $\{\eta^k\}$, $\{x^{k+1/2}\}$, and $\{v^k\}$. This leads to an HPR Algorithm 4. We show the equivalence of Algorithm 2, Algorithm 3, and Algorithm 4 in Proposition 3.1.

Algorithm 4 An HPR algorithm for solving COP (3.1)

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

Proposition 3.1. *Suppose that Assumptions 1 and 2 hold. Consider operators $\mathcal{G}_1$ and $\mathcal{G}_2$ defined in (3.9). Then the following equivalence holds:*

1. *If $\eta^0 \in \mathbb{X}$ in Algorithm 2 is identical to the η^0 in Algorithm 3, then the sequence $\{(x^{k+1/2}, x^{k+1}, v^{k+1}, \eta^{k+1})\}$ generated by Algorithm 2 coincides with the sequence $\{(x^{k+1/2}, x^{k+1}, v^{k+1}, \eta^{k+1})\}$ generated by Algorithm 3.*
2. *Starting from the same initial points $y^0 \in \text{dom}(f_1)$ and $x^0 \in \mathbb{X}$, the sequence $\{(z^{k+1}, x^{k+1/2}, y^{k+1}, x^{k+1})\}$ generated by Algorithm 3 coincides with the sequence $\{(z^{k+1}, x^{k+1/2}, y^{k+1}, x^{k+1})\}$ generated by Algorithm 4.*

Proof. We prove the first statement in the proposition by induction. Given η^0 in $\mathbb{X}$.

For $k = 0$, by the definition of z^1 in Algorithm 3, we have

$$0 \in \partial f_2(z^1) + B_2^* \eta^0 + \sigma B_2^* B_2 z^1.$$

Thus,

$$-B_2^*(\eta^0 + \sigma B_2 z^1) \in \partial f_2(z^1).$$

It follows from (Rockafellar, 1970, Theorem 23.5) that

$$z^1 \in \partial f_2^*(-B_2^*(\eta^0 + \sigma B_2 z^1)).$$

Hence,

$$-\sigma B_2 z^1 \in -\sigma B_2 \partial f_2^*(-B_2^*(\eta^0 + \sigma B_2 z^1)).$$

This means

$$\eta^0 \in \eta^0 + \sigma B_2 z^1 + \sigma \partial(f_2^* \circ (-B_2^*))(\eta^0 + \sigma B_2 z^1).$$

Since $\mathcal{G}_2(x) = \partial(f_2^* \circ (-B_2^*x))$ for $x \in \mathbb{X}$, we have

$$\eta^0 \in \eta^0 + \sigma B_2 z^1 + \sigma \mathcal{G}_2(\eta^0 + \sigma B_2 z^1),$$

which implies

$$x^{1/2} := \mathcal{J}_{\sigma \mathcal{G}_2}(\eta^0) = \eta^0 + \sigma B_2 z^1. \quad (3.11)$$

Similarly, by the definition of y^1 , we have

$$0 \in \partial f_1(y^1) + B_1^*(\eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c)).$$

It follows from (Rockafellar, 1970, Theorem 23.5) that

$$y^1 \in \partial f_1^*(-B_1^*(\eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c))).$$

Hence,

$$-\sigma B_1 y^1 \in -\sigma B_1 \partial f_1^*(-B_1^*(\eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c))).$$

This implies that

$$2(\eta^0 + \sigma B_2 z^1) - \eta^0 \in 2(\eta^0 + \sigma B_2 z^1) - \eta^0 + \sigma B_1 y^1 - \sigma B_1 \partial f_1^*(-B_1^*(\eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c))).$$

That is

$$\begin{aligned} 2(\eta^0 + \sigma B_2 z^1) - \eta^0 \in & \eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c) + \dots \\ & \sigma(\partial(f_1^* \circ (-B_1^*))(\eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c)) + c). \end{aligned}$$

Since $\mathcal{G}_1(x) = \partial(f_1^* \circ (-B_1^*x)) + c$ for $x \in \mathbb{X}$, we have

$$2(\eta^0 + \sigma B_2 z^1) - \eta^0 \in \eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c) + \sigma \mathcal{G}_1(\eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c)).$$

which implies

$$x^1 := \mathcal{J}_{\sigma\mathcal{G}_1}(2x^{1/2} - \eta^0) = \eta^0 + 2\sigma B_2 z^1 + \sigma(B_1 y^1 - c). \quad (3.12)$$

From (3.11) and (3.12), we have

$$x^1 - x^{1/2} = \sigma(B_1 y^1 + B_2 z^1 - c).$$

Hence

$$v^1 := \eta^0 + 2(x^1 - x^{1/2}) = \eta^0 + 2\sigma(B_1 y^1 + B_2 z^1 - c). \quad (3.13)$$

It follows that the update of η^1 is the same in Algorithm 2 as in Algorithm 3. Hence, the statement holds for $k = 0$. Assume that the statement holds for some $k \geq 1$. For the case $k + 1$, we can prove that the statement holds similarly to the case $k = 0$. Thus, by induction, the statement holds for all $k \geq 0$. This completes the first part.

Now, we show the proof of the second part by induction. Let $\eta^0 = \tilde{x}^0 + \sigma(B_1 y^0 - c)$.

For $k = 0$, from Algorithm 3, we have

$$\begin{aligned} z^1 &:= \arg \min_{z \in \mathbb{Z}} \{f_2(z) + \langle \eta^0, B_2 z \rangle + \frac{\sigma}{2} \|B_2 z\|^2\} \\ &= \arg \min_{z \in \mathbb{Z}} \{f_2(z) + \langle \tilde{x}^0 + \sigma(B_1 y^0 - c), B_2 z \rangle + \frac{\sigma}{2} \|B_2 z\|^2\} \\ &= \arg \min_{z \in \mathbb{Z}} \{f_2(z) + \langle \tilde{x}^0, B_1 y^0 + B_2 z - c \rangle + \frac{\sigma}{2} \|B_1 y^0 + B_2 z - c\|^2\}. \end{aligned}$$

For $x^{1/2}$, from Algorithm 3 and definition of η^0 , we have

$$x^{1/2} := \eta^0 + \sigma B_2 z^1 = \tilde{x}^0 + \sigma(B_1 y^0 + B_2 z^1 - c).$$

For y^1 , from Algorithm 3, we have

$$\begin{aligned} y^1 &:= \arg \min_{y \in \mathbb{Y}} \{f_1(y) + \langle \eta^0 + 2\sigma B_2 z^1, B_1 y - c \rangle + \frac{\sigma}{2} \|B_1 y - c\|^2\} \\ &= \arg \min_{y \in \mathbb{Y}} \{f_1(y) + \langle \tilde{x}^0 + \sigma(B_1 y^0 - c) + 2\sigma B_2 z^1, B_1 y - c \rangle + \frac{\sigma}{2} \|B_1 y - c\|^2\} \\ &= \arg \min_{y \in \mathbb{Y}} \{f_1(y) + \langle \tilde{x}^0 + \sigma(B_1 y^0 + B_2 z^1 - c), B_1 y + B_2 z^1 - c \rangle + \frac{\sigma}{2} \|B_1 y + B_2 z^1 - c\|^2\} \\ &= \arg \min_{y \in \mathbb{Y}} \{f_1(y) + \langle x^{1/2}, B_1 y + B_2 z^1 - c \rangle + \frac{\sigma}{2} \|B_1 y + B_2 z^1 - c\|^2\}. \end{aligned}$$

For x^1 , from Algorithm 3, we have

$$\begin{aligned}
x^1 &:= \eta^0 + \sigma(B_1 y^1 - c) + 2\sigma B_2 z^1 \\
&= \tilde{x}^0 + \sigma(B_1 y^0 - c) + \sigma(B_1 y^1 - c) + 2\sigma B_2 z^1 \\
&= \tilde{x}^0 + \sigma(B_1 y^0 + B_2 z^1 - c) + \sigma(B_1 y^1 + B_2 z^1 - c) \\
&= x^{1/2} + \sigma(B_1 y^1 + B_2 z^1 - c).
\end{aligned}$$

For $\tilde{x}^1$ defined in (3.10), We have

$$\begin{aligned}
\tilde{x}^1 &:= \eta^1 - \sigma(B_1 y^1 - c) \\
&= \frac{1}{2}\eta^0 + \frac{1}{2}v^1 - \sigma(B_1 y^1 - c) \\
&= \frac{1}{2}(\tilde{x}^0 + \sigma(B_1 y^0 - c)) + \frac{1}{2}(\tilde{x}^0 + \sigma(B_1 y^0 - c) + 2\sigma(B_1 y^1 + B_2 z^1 - c)) - \sigma(B_1 y^1 - c) \\
&= \frac{1}{2}(\tilde{x}^0 + \sigma(B_1 y^0 - c)) + \frac{1}{2}(x^1) - \frac{1}{2}\sigma(B_1 y^1 - c).
\end{aligned}$$

Hence, the statement holds for $k = 0$. Assume that the statement holds for some $k \geq 0$. Then, we can prove the statement for $k + 1$ similarly to the case for $k = 0$. Thus, by induction, we have completed the proof. $\square$

3.1.1 Convergence Analysis

Now, we prove the convergence of Algorithm 4 for solving COP (3.1).

Corollary 3.3. *Assume that Assumption 1 and Assumption 2 hold. Let $\{(y^k, z^k, x^k)\}$ and $\{(\tilde{x}^k, x^{k+1/2})\}$ be the sequences generated by Algorithm 4. Then, we have*

$$y^k \rightarrow y^*, \quad z^k \rightarrow z^*, \quad x^k \rightarrow x^*, \quad x^{k+1/2} \rightarrow x^*, \quad \text{and} \quad \tilde{x}^k \rightarrow x^*,$$

where (y^*, z^*) is a solution to the problem (3.1) and x^* is a solution to the problem (3.2).

Proof. First of all, Assumption 1 ensures the existence of (y^*, z^*, x^*) (Rockafellar, 1970, Corollary 28.2.2). Moreover, Assumption 2 ensures that both $\Sigma_{f_1} + \sigma B_1^* B_1$ and $\Sigma_{f_2} + \sigma B_2^* B_2$ are positive definite for any $\sigma > 0$. As a result, the subproblems of Algorithm 4 are all solvable. Given any $y^0 \in \text{dom}(f_1)$ and $x^0 \in \mathbb{X}$, let the sequence $\{(x^{k+1/2}, v^{k+1}, \eta^{k+1})\}$ be generated by Algorithm 3 with $\eta^0 = x^0 + \sigma(B_1 y^0 - c)$.

From Proposition 3.1 and Corollary 3.1, we know the sequence $\{(x^{k+1/2}, x^{k+1})\}$ is convergent. Next, we prove the convergence of $\{z^k\}$. According to Algorithm 3 and Proposition 3.1, we have for all $k \geq 0$,

$$z^{k+1} = \arg \min_z \left\{ f_2(z) + \langle \eta^k, B_2 z \rangle + \frac{\sigma}{2} \|B_2 z\|^2 \right\}. \quad (3.14)$$

Denote $\hat{f}_2(z) := f_2(z) + \frac{\sigma}{2} \|B_2 z\|^2$, which is a strongly convex function. Thus, $\hat{f}_2^*$ is essentially smooth (Rockafellar, 1970, Theorem 26.3). The first-order optimality condition of (3.14) implies

$$0 \in \partial \hat{f}_2(z^{k+1}) + B_2^* \eta^k \quad \forall k \geq 0. \quad (3.15)$$

Since $\hat{f}_2$ is a proper closed convex function, by (Rockafellar, 1970, Theorem 23.5), (3.15) is equivalent to

$$z^{k+1} = \nabla \hat{f}_2^*(-B_2^* \eta^k) \quad \forall k \geq 0. \quad (3.16)$$

It follows from the convergence of $\{\eta^k\}$ by Corollary 3.1 and the continuity of $\nabla \hat{f}_2^*$ (Rockafellar, 1970, Theorem 25.5) that $\{z^k\}$ is convergent. Similarly, we can obtain the convergence of $\{y^k\}$. As a result, the convergence of $\{y^k\}$ and $\{x^k\}$ yields the convergence of $\{\tilde{x}^k\}$.

Assume that (y^*, z^*, x^*) is the limit point of sequence $\{y^k, z^k, x^k\}$. Since for all $k \geq 0$,

$$v^{k+1} = \eta^k + 2\sigma(B_1 y^{k+1} + B_2 z^{k+1} - c)$$

from Algorithm 3, we can obtain

$$B_1 y^* + B_2 z^* = c \quad (3.17)$$

from Corollary 3.1 by taking the limit. It follows that $\lim_{k \rightarrow \infty} x^{k+1/2} = \lim_{k \rightarrow \infty} \tilde{x}^k = x^*$. By Algorithm 3, we have

$$-B_2^* x^{k+1/2} \in \partial f_2(z^{k+1}), \quad -B_1^* x^{k+1} \in \partial f_1(y^{k+1}) \quad \forall k \geq 0.$$

It follows from Corollary 3.1 and (Rockafellar, 1970, Theorem 24.4) that

$$-B_2^*x^* \in \partial f_2(z^*), \quad -B_1^*x^* \in \partial f_1(y^*). \quad (3.18)$$

Together with (3.17), we have

$$-B_2^*x^* \in \partial f_2(z^*), \quad -B_1^*x^* \in \partial f_1(y^*), \quad B_1y^* + B_2z^* - c = 0.$$

This completes the proof (Rockafellar, 1970, Corollary 28.3.1). $\square$

Remark 3.1. *Assumption 2 holds automatically if either B_i is injective or f_i is strongly convex (for $i = 1, 2$). Han et al. (2018) gives an example where the sufficient conditions just mentioned fail to hold, but Assumption 2 still holds.*

3.1.2 Iteration Complexity Analysis

In this subsection, we analyze the iteration complexity of Algorithm 4 for solving the optimization problem (3.1). Given the sequence $\{(y^k, z^k)\}$ generated by Algorithm 4, define

$$h(y^{k+1}, z^{k+1}) := f_1(y^{k+1}) + f_2(z^{k+1}) - f_1(y^*) - f_2(z^*) \quad \forall k \geq 0, \quad (3.19)$$

where (y^*, z^*) is the limit point of the sequence $\{(y^k, z^k)\}$. We also define

$$d_{f_1}(x) := f_1^*(-B_1^*x) + \langle x, c \rangle \quad \text{and} \quad d_{f_2}(x) := f_2^*(-B_2^*x), \quad \forall x \in \mathbb{X}.$$

Furthermore, to analyze the iteration complexity of the HPR algorithm in terms of KKT residual, we define the following residual mapping associated with the KKT system:

$$\mathcal{R}(y, z, x) = \begin{pmatrix} y - \text{Prox}_{f_1}(y - B_1^*x) \\ z - \text{Prox}_{f_2}(z - B_2^*x) \\ c - B_1y - B_2z \end{pmatrix} \quad \forall (y, z, x) \in \mathbb{Y} \times \mathbb{Z} \times \mathbb{X}. \quad (3.20)$$

Then (y^*, z^*, x^*) satisfies the KKT system (3.3) if and only if $\mathcal{R}(y^*, z^*, x^*) = 0$.

We first prove the following lemma about the objective error before establishing the iteration complexity of the HPR algorithm.

Lemma 3.1. *Suppose that Assumption 1 and Assumption 2 hold. Consider operators $\mathcal{G}_1$ and $\mathcal{G}_2$ defined in (3.9). Let $\{(y^{k+1}, z^{k+1})\}$ be the sequence generated by Algorithm 4 and the sequence $\{(x^{k+1/2}, x^{k+1}, v^{k+1}, \eta^{k+1})\}$ be generated by Algorithm 3 with $\eta^0 = \tilde{x}^0 + \sigma(B_1 y^0 - c)$. Then, we have for all $k \geq 0$,*

$$4\sigma h(y^{k+1}, z^{k+1}) = -4\sigma (d_{f_1}(x^{k+1}) + d_{f_2}(x^{k+1/2}) - d_{f_1}(x^*) - d_{f_2}(x^*)) + \|\eta^k - v^{k+1}\|^2 + 2\langle \eta^k - v^{k+1}, v^{k+1} \rangle. \quad (3.21)$$

Moreover, the following bounds hold for all $k \geq 0$:

$$4\sigma h(y^{k+1}, z^{k+1}) \leq \|\eta^k - (\eta^* - x^*)\|^2 - \|v^{k+1} - (\eta^* - x^*)\|^2, \quad (3.22)$$

and

$$h(y^{k+1}, z^{k+1}) \geq \langle B_1 y^{k+1} + B_2 z^{k+1} - c, -x^* \rangle, \quad (3.23)$$

where $\eta^* = \Pi_{\text{Fix } \mathcal{F}_\sigma^{\text{PR}}}(\eta^0)$ and $x^* = \mathcal{J}_{\sigma \mathcal{G}_2}(\eta^*)$.

Proof. We first prove the equation (3.21). For all $k \geq 0$, from Algorithm 3 we have

$$-B_2^* x^{k+1/2} \in \partial f_2(z^{k+1}), \quad -B_1^* x^{k+1} \in \partial f_1(y^{k+1}). \quad (3.24)$$

It follows from (Rockafellar, 1970, Theorem 23.5) that for all $k \geq 0$,

$$\begin{cases} d_{f_1}(x^{k+1}) = \langle x^{k+1}, c \rangle - \langle B_1^* x^{k+1}, y^{k+1} \rangle - f_1(y^{k+1}), \\ d_{f_2}(x^{k+1/2}) = -\langle B_2^* x^{k+1/2}, z^{k+1} \rangle - f_2(z^{k+1}). \end{cases}$$

Therefore, for all $k \geq 0$,

$$\begin{aligned} & -d_{f_1}(x^{k+1}) - d_{f_2}(x^{k+1/2}) \\ &= f_1(y^{k+1}) + f_2(z^{k+1}) - \langle x^{k+1}, c \rangle + \langle B_1^* x^{k+1}, y^{k+1} \rangle + \langle B_2^* x^{k+1/2}, z^{k+1} \rangle \\ &= f_1(y^{k+1}) + f_2(z^{k+1}) + \langle x^{k+1}, B_1 y^{k+1} + B_2 z^{k+1} - c \rangle - \langle x^{k+1} - x^{k+1/2}, B_2 z^{k+1} \rangle. \end{aligned}$$

From Algorithm 3, we also have for all $k \geq 0$,

$$x^{k+1} - x^{k+1/2} = \sigma (B_1 y^{k+1} + B_2 z^{k+1} - c) = \frac{1}{2}(v^{k+1} - \eta^k).$$

Thus, for all $k \geq 0$,

$$-d_{f_1}(x^{k+1}) - d_{f_2}(x^{k+1/2}) = f_1(y^{k+1}) + f_2(z^{k+1}) + \frac{1}{\sigma} \langle x^{k+1} - \sigma B_2 z^{k+1}, x^{k+1} - x^{k+1/2} \rangle. \quad (3.25)$$

Recall that $x^{k+1} - \sigma B_2 z^{k+1} = \eta^k + \sigma(B_1 y^{k+1} + B_2 z^{k+1} - c) = \eta^k + x^{k+1} - x^{k+1/2}$.

Therefore, for all $k \geq 0$,

$$x^{k+1} - \sigma B_2 z^{k+1} = \eta^k + \frac{1}{2}(v^{k+1} - \eta^k) = \frac{1}{2}(\eta^k - v^{k+1}) + v^{k+1},$$

and

$$\begin{aligned} \frac{1}{\sigma} \langle x^{k+1} - \sigma B_2 z^{k+1}, x^{k+1} - x^{k+1/2} \rangle &= \frac{1}{\sigma} \left\langle \frac{1}{2}(\eta^k - v^{k+1}) + v^{k+1}, \frac{1}{2}(v^{k+1} - \eta^k) \right\rangle \\ &= -\frac{1}{4\sigma} \|v^{k+1} - \eta^k\|^2 - \frac{1}{2\sigma} \langle \eta^k - v^{k+1}, v^{k+1} \rangle. \end{aligned} \quad (3.26)$$

According to Corollary 3.1, x^* is a solution to problem (3.4) and problem (3.2) under Assumption 1. We also have the limit point (y^*, z^*) of the sequence $\{(y^k, z^k)\}$ is a solution to problem (3.1) from Corollary 3.3. It follows from (Rockafellar, 1970, Theorem 28.4) that

$$-d_{f_1}(x^*) - d_{f_2}(x^*) = f_1(y^*) + f_2(z^*). \quad (3.27)$$

Therefore, it follows from (3.25), (3.26), and (3.27) that the equation (3.21) holds.

Next, we prove the upper bound of $h(y^{k+1}, z^{k+1})$ for all $k \geq 0$. According to Algorithm 2 and Corollary 3.1 that $(\eta^* - x^*)/\sigma \in \partial d_{f_2}(x^*)$ and $(x^* - \eta^*)/\sigma \in \partial d_{f_1}(x^*)$. Hence,

$$\begin{cases} d_{f_1}(x^{k+1}) - d_{f_1}(x^*) \geq \langle (x^* - \eta^*)/\sigma, x^{k+1} - x^* \rangle, \\ d_{f_2}(x^{k+1/2}) - d_{f_2}(x^*) \geq \langle (\eta^* - x^*)/\sigma, x^{k+1/2} - x^* \rangle. \end{cases}$$

Thus,

$$\begin{aligned} &-4\sigma (d_{f_1}(x^{k+1}) + d_{f_2}(x^{k+1/2}) - d_{f_1}(x^*) - d_{f_2}(x^*)) \\ &\leq 4 \langle x^{k+1} - x^{k+1/2}, \eta^* - x^* \rangle = 2 \langle v^{k+1} - \eta^k, \eta^* - x^* \rangle. \end{aligned}$$

Together with (3.21), we obtain the inequality (3.22).

Finally, according to (3.3) and Corollary 3.1, we have for all $k \geq 0$,

$$f_1(y^{k+1}) - f_1(y^*) \geq \langle -B_1^* x^*, y^{k+1} - y^* \rangle, \quad f_2(z^{k+1}) - f_2(z^*) \geq \langle -B_2^* x^*, z^{k+1} - z^* \rangle.$$

Thus, the inequality (3.23) holds since $B_1 y^* + B_2 z^* = c$. This completes the proof of the lemma. $\square$

Now, we are ready to prove the iteration complexity of the HPR algorithm for solving the optimization problem (3.1).

Theorem 3.1. *Suppose that Assumption 1 and Assumption 2 hold. Consider operators $\mathcal{G}_1$ and $\mathcal{G}_2$ defined in (3.9). Let $\{(y^{k+1}, z^{k+1}, x^{k+1})\}$ be the sequence generated by Algorithm 4. Let (y^*, x^*) be the limit point of $\{(y^{k+1}, x^{k+1})\}$. Then for all $k \geq 0$, we have the following bounds:*

$$\|\mathcal{R}(y^{k+1}, z^{k+1}, x^{k+1})\| \leq \frac{1}{k+1} \left(\frac{\sigma \|B_2^*\| + 1}{\sigma} (\|x^0 - x^*\| + \sigma \|B_1 y^0 - B_1 y^*\|) \right). \quad (3.28)$$

Moreover, we have for $k \geq 0$ that

$$h(y^{k+1}, z^{k+1}) \geq -\frac{1}{k+1} \left(\frac{\|x^*\|}{\sigma} (\|x^0 - x^*\| + \sigma \|B_1 y^0 - B_1 y^*\|) \right)$$

and

$$\begin{aligned} & h(y^{k+1}, z^{k+1}) \\ & \leq \frac{1}{k+1} \left(\frac{(\|x^0 - x^*\| + \sigma \|B_1 y^0 - B_1 y^*\|)^2 + \|x^*\| (\|x^0 - x^*\| + \sigma \|B_1 y^0 - B_1 y^*\|)}{\sigma} \right). \end{aligned}$$

Proof. Take the sequence $\{(x^{k+1}, v^{k+1}, \eta^{k+1})\}$ generated by Algorithm 3 with $\eta^0 = x^0 + \sigma(B_1 y^0 - c)$. From Corollary 3.1, Proposition 3.1, and Corollary 3.3, we have the sequence $\{\eta^{k+1}\}$ converges to $\eta^* := \Pi_{\text{Fix } \mathcal{F}_\sigma^{\text{PR}}}(\eta^0)$, and $\eta^* = x^* + \sigma(B_1 y^* - c)$. Hence, we have

$$\|\eta^0 - \eta^*\| = \|x^0 - x^* + \sigma(B_1 y^0 - B_1 y^*)\|.$$

It together with the iteration complexity of Halpern iteration in Corollary 3.2 implies that

$$\|v^{k+1} - \eta^k\| \leq \frac{2\|\eta^0 - \eta^*\|}{k+1} = \frac{2\|x^0 - x^* + \sigma(B_1y^0 - B_1y^*)\|}{k+1} \quad \forall k \geq 0. \quad (3.29)$$

According to Algorithm 3 and Proposition 3.1, we have

$$\|(B_1y^{k+1} + B_2z^{k+1} - c)\| = \frac{1}{2\sigma}\|v^{k+1} - \eta^k\| \leq \frac{\|x^0 - x^* + \sigma(B_1y^0 - B_1y^*)\|}{\sigma(k+1)} \quad \forall k \geq 0. \quad (3.30)$$

Moreover, from Algorithm 3, we also have for all $k \geq 0$,

$$-B_2^*x^{k+1/2} \in \partial f_2(z^{k+1}), \quad -B_1^*x^{k+1} \in \partial f_1(y^{k+1}),$$

and

$$x^{k+1} = x^{k+1/2} + \sigma(B_1y^{k+1} + B_2z^{k+1} - c).$$

This yields that for all $k \geq 0$

$$y^{k+1} - \text{Prox}_{f_1}(y^{k+1} - B_1^*x^{k+1}) = 0$$

and

$$\begin{aligned} & \|z^{k+1} - \text{Prox}_{f_2}(z^{k+1} - B_2^*x^{k+1})\| \\ &= \|\text{Prox}_{f_2}(z^{k+1} - B_2^*x^{k+1/2}) - \text{Prox}_{f_2}(z^{k+1} - B_2^*x^{k+1})\| \\ &\leq \|\sigma B_2^*(B_1y^{k+1} + B_2z^{k+1} - c)\| \\ &\leq \sigma\|B_2^*\|\|B_1y^{k+1} + B_2z^{k+1} - c\|. \end{aligned}$$

As a result, for $k \geq 0$, we have

$$\begin{aligned} \|\mathcal{R}(y^{k+1}, z^{k+1}, x^{k+1})\| &\stackrel{(3.30)}{\leq} \sqrt{\sigma^2\|B_2^*\|^2 + 1}\|B_1y^{k+1} + B_2z^{k+1} - c\| \\ &\leq (\sigma\|B_2^*\| + 1) \frac{\|x^0 - x^* + \sigma(B_1y^0 - B_1y^*)\|}{\sigma(k+1)} \\ &\leq \frac{1}{k+1} \left(\frac{\sigma\|B_2^*\| + 1}{\sigma} (\|x^0 - x^*\| + \sigma\|B_1y^0 - B_1y^*\|) \right). \end{aligned}$$

Next, we show the iteration complexity regarding the objective error. We will first show that

$$\|v^k - \eta^*\| \leq \|\eta^0 - \eta^*\| \quad \forall k \geq 1. \quad (3.31)$$

We prove it by induction. When $k = 1$, we have $\|v^1 - \eta^*\| \leq \|\eta^0 - \eta^*\|$ by the nonexpansiveness of $\mathcal{F}_\sigma^{\text{PR}}$. Assume (3.31) holds for some $k \geq 1$. Then, we have

$$\begin{aligned}\|v^{k+1} - \eta^*\| &\leq \|\eta^k - \eta^*\| \\ &= \left\| \frac{1}{k+1} \eta^0 + \frac{k}{k+1} v^k - \eta^* \right\| \\ &\leq \frac{1}{k+1} \|\eta^0 - \eta^*\| + \frac{k}{k+1} \|v^k - \eta^*\| \\ &\leq \|\eta^0 - \eta^*\|,\end{aligned}$$

which proves (3.31). It follows that

$$\|v^{k+1} - (\eta^* - x^*)\| \leq \|\eta^0 - \eta^*\| + \|x^*\| \quad \forall k \geq 0. \quad (3.32)$$

Similarly, we have

$$\|\eta^k - (\eta^* - x^*)\| \leq \|\eta^0 - \eta^*\| + \|x^*\| \quad \forall k \geq 0. \quad (3.33)$$

From (3.22), we have for all $k \geq 0$,

$$\begin{aligned}& h(y^{k+1}, z^{k+1}) \\ & \leq \frac{1}{4\sigma} (\|\eta^k - (\eta^* - x^*)\|^2 - \|v^{k+1} - (\eta^* - x^*)\|^2) \\ & = \frac{1}{4\sigma} \langle \eta^k + v^{k+1} - 2(\eta^* - x^*), \eta^k - v^{k+1} \rangle \\ & \leq \frac{1}{4\sigma} (\|\eta^k - (\eta^* - x^*)\| + \|v^{k+1} - (\eta^* - x^*)\|) \|\eta^k - v^{k+1}\| \\ & \stackrel{(3.32)(3.33)}{\leq} \frac{1}{2\sigma} (\|\eta^0 - \eta^*\| + \|x^*\|) \|\eta^k - v^{k+1}\| \\ & \stackrel{(2.9)}{\leq} \frac{(\|x^0 - x^* + \sigma(B_1 y^0 - B_1 y^*)\| + \|x^*\|) \|x^0 - x^* + \sigma(B_1 y^0 - B_1 y^*)\|}{\sigma(k+1)} \\ & \leq \frac{1}{k+1} \left(\frac{(\|x^0 - x^*\| + \sigma\|B_1 y^0 - B_1 y^*\|)^2 + \|x^*\|(\|x^0 - x^*\| + \sigma\|B_1 y^0 - B_1 y^*\|)}{\sigma} \right).\end{aligned}$$

For the lower bound, from (3.23) and Algorithm 3, we have for all $k \geq 0$,

$$\begin{aligned}h(y^{k+1}, z^{k+1}) &\geq \langle B_1 y^{k+1} + B_2 z^{k+1} - c, -x^* \rangle \\ &= \frac{1}{2\sigma} \langle v^{k+1} - \eta^k, -x^* \rangle \\ &\geq -\frac{1}{2\sigma} \|v^{k+1} - \eta^k\| \|x^*\| \\ &\stackrel{(2.9)}{\geq} -\frac{\|x^0 - x^* + \sigma(B_1 y^0 - B_1 y^*)\| \|x^*\|}{\sigma(k+1)} \\ &\geq -\frac{1}{k+1} \left(\frac{\|x^*\|}{\sigma} (\|x^0 - x^*\| + \sigma\|B_1 y^0 - B_1 y^*\|) \right).\end{aligned}$$

This completes the proof. $\square$

Remark 3.2. *If all the subproblems in Algorithm 4 are solvable, the $O(1/k)$ iteration complexity in terms of the KKT residual and the objective error still holds without Assumption 2. This assumption is only for the convergence of the sequence $\{(y^k, z^k)\}$.*

3.2 An HPR Method for Wasserstein Barycenter Problems

We can write the WBP with pre-specified finite supports defined in (1.5) as the following standard form of an LP problem:

$$\begin{aligned} \min_x \quad & \langle c, x \rangle + \delta_K(x) \\ \text{s.t.} \quad & \hat{A}x = \hat{b}, \end{aligned} \tag{3.34}$$

where

1. $x = (\text{vec}(\pi^1); \dots; \text{vec}(\pi^T); \mathbf{a}^c)$, K is the nonnegative orthant with compatible dimension,
2. $\hat{b} = (\mathbf{a}^1; \mathbf{a}^2; \dots; \mathbf{a}^T; \mathbf{0}_m; \dots; \mathbf{0}_m; 1)$, $c = (\text{vec}(D^1); \dots; \text{vec}(D^T); \mathbf{0}_m)$,
3. $\hat{A} = \begin{bmatrix} A_1 & \mathbf{0} \\ \hat{A}_2 & \hat{A}_3 \\ \mathbf{0} & \mathbf{1}_m^\top \end{bmatrix}$, $A_1 = \text{diag}(I_{m_1} \otimes \mathbf{1}_m^\top, \dots, I_{m_T} \otimes \mathbf{1}_m^\top)$,
 $\hat{A}_2 = \text{diag}(\mathbf{1}_{m_1}^\top \otimes I_m, \dots, \mathbf{1}_{m_T}^\top \otimes I_m)$, and $\hat{A}_3 = -\mathbf{1}_T \otimes I_m$.

Define

$$M := \sum_{i=1}^T m_i + T(m-1) + 1, \quad N := m \sum_{i=1}^T m_i + m. \tag{3.35}$$

Let A_2 and A_3 be the matrices which are obtained from $\hat{A}_2$ and $\hat{A}_3$ by removing the 1-st, $(m+1)$ -th, $\dots$, $(T-1)m+1$ -th rows of $\hat{A}_2$ and $\hat{A}_3$, respectively. That is

$$A_2 = \text{diag}(\mathbf{1}_{m_1}^\top \otimes [\mathbf{0}_{m-1}, I_{m-1}], \dots, \mathbf{1}_{m_t}^\top \otimes [\mathbf{0}_{m-1}, I_{m-1}]), \text{ and } A_3 = -\mathbf{1}_T \otimes [\mathbf{0}_{m-1}, I_{m-1}].$$

Define

$$A := \begin{bmatrix} A_1 & \mathbf{0} \\ A_2 & A_3 \\ \mathbf{0} & \mathbf{1}_m^\top \end{bmatrix} \in \mathbb{R}^{M \times N}, \quad b := (\mathbf{a}^1; \mathbf{a}^2; \dots; \mathbf{a}^T; \mathbf{0}_{m-1}; \dots; \mathbf{0}_{m-1}; 1) \in \mathbb{R}^M. \quad (3.36)$$

Ge et al. (2019) made the following useful observation:

Proposition 3.2 (Ge et al. (2019, Lemma 3.1)). *Consider $A \in \mathbb{R}^{M \times N}$ and $b \in \mathbb{R}^M$ defined in (3.36). Then A has full row rank, and $\{x \in \mathbb{R}^N \mid Ax = b\} = \{x \in \mathbb{R}^N \mid \hat{A}x = \hat{b}\}$.*

As a result, the linear programming problem (3.34) is equivalent to

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & \langle c, x \rangle + \delta_K(x) \\ \text{s.t.} \quad & Ax = b. \end{aligned} \quad (3.37)$$

The dual problem of (3.37) is

$$\min_{y \in \mathbb{R}^M, z \in \mathbb{R}^N} \{-\langle b, y \rangle + \delta_K^*(-z) \mid A^*y + z = c\}, \quad (3.38)$$

where $\delta_K^*(\cdot)$ is the support function of K . The KKT conditions associated with (3.37) and (3.38) are given by

$$A^*y + z = c, \quad Ax = b, \quad K \ni x \perp z \in K, \quad (3.39)$$

where $x \perp z$ means x is perpendicular to z , i.e., $\langle x, z \rangle = 0$.

An HPR for solving the WBP (3.38) is presented in Algorithm 5, which is a direct application of Algorithm 4.

Algorithm 5 An HPR algorithm for solving the WBP (3.38)

- 1: **Input:** $x^0 \in \mathbb{R}^N$, $y^0 \in \mathbb{R}^M$, and $\sigma > 0$
- 2: **Initialization:** $\tilde{x}^0 := x^0$, $z^0 := c - A^*y^0$
- 3: **for** $k = 0, 1, \dots$ **do**
- 4: Step 1. $z^{k+1} = \Pi_K(c - A^*y^k - \frac{1}{\sigma}\tilde{x}^k)$
- 5: Step 2. $x^{k+1/2} = \tilde{x}^k + \sigma(z^{k+1} + A^*y^k - c)$
- 6: Step 3. Compute y^{k+1} by solving

$$AA^*y = \frac{1}{\sigma}b - A \left(\frac{1}{\sigma}x^{k+1/2} + z^{k+1} - c \right) \quad (3.40)$$

- 7: Step 4. $x^{k+1} = x^{k+1/2} + \sigma(z^{k+1} + A^*y^{k+1} - c)$
 - 8: Step 5. $\tilde{x}^{k+1} = \left(\frac{1}{k+2}\tilde{x}^0 + \frac{k+1}{k+2}x^{k+1} \right) + \frac{\sigma}{k+2}[(A^*y^0 - c) - (A^*y^{k+1} - c)]$
-

Next, we will propose an efficient procedure for solving $AA^*y = R$, where A is the matrix defined in (3.36) and $R \in \mathbb{R}^M$ is any given vector. The computational complexity for solving the linear system is only $O\left(Tm + \sum_{t=1}^T m_t\right)$. For notation convenience, we denote $M_1 := \sum_{t=1}^T m_t$ and $M_2 := T(m-1)$. By direct calculations, AA^* can be written in the following form:

$$AA^* = \begin{bmatrix} A_1 & \mathbf{0} \\ A_2 & A_3 \\ \mathbf{0} & \mathbf{1}_m^\top \end{bmatrix} \begin{bmatrix} A_1^* & A_2^* & \mathbf{0} \\ \mathbf{0} & A_3^* & \mathbf{1}_m \end{bmatrix} = \begin{bmatrix} E_1 & E_2 & \mathbf{0} \\ E_2^* & E_3 + E_4 & E_5 \\ \mathbf{0} & E_5^* & m \end{bmatrix}, \quad (3.41)$$

where

1. $E_1 := A_1 A_1^* = \text{diag}(mI_{m_1}, \dots, mI_{m_T}) \in \mathbb{R}^{M_1 \times M_1}$,
2. $E_2 := A_1 A_2^* = \text{diag}(\mathbf{1}_{m_1} \mathbf{1}_{m-1}^\top, \dots, \mathbf{1}_{m_T} \mathbf{1}_{m-1}^\top) \in \mathbb{R}^{M_1 \times M_2}$,
3. $E_3 := A_2 A_2^* = \text{diag}(m_1 I_{m-1}, \dots, m_T I_{m-1}) \in \mathbb{R}^{M_2 \times M_2}$,
4. $E_4 := A_3 A_3^* = (\mathbf{1}_T \mathbf{1}_T^\top) \otimes I_{m-1} \in \mathbb{R}^{M_2 \times M_2}$,
5. $E_5 := A_3 \mathbf{1}_m = -\mathbf{1}_T \otimes \mathbf{1}_{m-1} \in \mathbb{R}^{M_2}$.

To better explore the structure of $AA^*y = R$, we rewrite it equivalently as

$$AA^*y = \begin{bmatrix} E_1 & E_2 & \mathbf{0} \\ E_2^* & E_3 + E_4 & E_5 \\ \mathbf{0} & E_5^* & m \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}. \quad (3.42)$$

where $y := (y_1; y_2; y_3) \in \mathbb{R}^{M_1} \times \mathbb{R}^{M_2} \times \mathbb{R}$ and $R := (R_1; R_2; R_3) \in \mathbb{R}^{M_1} \times \mathbb{R}^{M_2} \times \mathbb{R}$.

To further explore the block structure of the linear system, we can denote $y_1 := (y_1^1; \dots; y_1^T) \in \mathbb{R}^{m_1} \times \dots \times \mathbb{R}^{m_T}$, $y_2 = (y_2^1; \dots; y_2^T) \in \mathbb{R}^{m-1} \times \dots \times \mathbb{R}^{m-1}$. Correspondingly, we write $R_1 = (R_1^1; \dots; R_1^T)$ and $R_2 = (R_2^1; \dots; R_2^T)$. The next proposition shows the solution y to the linear system (3.42).

Proposition 3.3. *Consider $A \in \mathbb{R}^{M \times N}$ defined in (3.36). Given $R \in \mathbb{R}^M$, the solution y to $AA^*y = R$ in the form (3.42) is given by*

$$y_2^t = \frac{1}{m_t} (\hat{y}_2^t - \hat{y}_2^a), \quad t = 1, \dots, T, \quad (3.43)$$

$$y_1^t = \frac{R_1^t}{m} - \frac{\mathbf{1}_{m-1}^\top y_2^t}{m} \mathbf{1}_{m_t}, \quad t = 1, \dots, T, \quad (3.44)$$

$$y_3 = \frac{1}{m} (R_3 + \mathbf{1}_{M_2}^\top y_2), \quad (3.45)$$

where

$$1. \hat{y}_2^t := R_2^t + (\mathbf{1}_{m-1}^\top R_2^t - \mathbf{1}_{m_t}^\top R_1^t + R_3) \mathbf{1}_{m-1}, \quad t = 1, \dots, T,$$

$$2. \hat{y}_2^a := \sum_{t=1}^T \frac{\bar{m}}{m_t} \hat{y}_2^t, \text{ and } \bar{m} := (1 + \sum_{t=1}^T \frac{1}{m_t})^{-1}.$$

Proof. By some direct calculations, we can solve (3.42) equivalently as

$$y_1^t = (E_1^{-1} (R_1 - E_2 y_2))^t = \frac{R_1^t}{m} - \frac{\mathbf{1}_{m-1}^\top y_2^t}{m} \mathbf{1}_{m_t}, \quad t = 1, \dots, T, \quad (3.46)$$

$$y_3 = \frac{1}{m} (R_3 - E_5^* y_2) = \frac{1}{m} (R_3 + \mathbf{1}_{M_2}^\top y_2), \quad (3.47)$$

$$(E_3 - E_2^* E_1^{-1} E_2 + E_4 - \frac{1}{m} E_5 E_5^*) y_2 = R_2 - E_2^* E_1^{-1} R_1 - \frac{1}{m} E_5 R_3. \quad (3.48)$$

As a result, the key is to obtain y_2 by solving (3.48). For convenience, denote $\hat{E}_3 := E_3 - E_2^*(E_1)^{-1}E_2$ and $\hat{E}_4 := E_4 - \frac{1}{m}E_5E_5^*$. Then, the linear system (3.48) can be rewritten as

$$(\hat{E}_3 + \hat{E}_4)y_2 = \hat{R}_2,$$

where $\hat{R}_2 := R_2 - E_2^*E_1^{-1}R_1 - \frac{1}{m}E_5R_3$. Define $Q := I_{m-1} - 1/m(\mathbf{1}_{m-1}\mathbf{1}_{m-1}^\top)$. By (3.41) and some simple calculations, we have

$$\hat{E}_4 = \mathbf{1}_T\mathbf{1}_T^\top \otimes Q, \quad (3.49)$$

and

$$\hat{E}_3 = \text{diag}(m_1Q, \dots, m_TQ).$$

Moreover, by the Sherman–Morrison–Woodbury formula, we directly get

$$\hat{E}_3^{-1} = \text{diag}(1/m_1Q^{-1}, \dots, 1/m_TQ^{-1}), \quad (3.50)$$

where $Q^{-1} = (I_{m-1} + \mathbf{1}_{m-1}\mathbf{1}_{m-1}^\top)$. Denote $\hat{Q} := Q^{1/2}$ such that $\hat{Q}\hat{Q} = Q$, and $\bar{Q} := \mathbf{1}_T \otimes \hat{Q}$. Then we have $\hat{E}_4 = \bar{Q}\bar{Q}^*$. Using the Sherman–Morrison–Woodbury formula, we can obtain

$$(\hat{E}_3 + \hat{E}_4)^{-1} = \hat{E}_3^{-1} - \hat{E}_3^{-1}\bar{Q}W\bar{Q}^*\hat{E}_3^{-1},$$

where $W := (I_{m-1} + \bar{Q}^*\hat{E}_3^{-1}\bar{Q})^{-1} = (I_{m-1} + \sum_{i=1}^T \hat{Q}(\hat{E}_3^i)^{-1}\hat{Q})^{-1}$. It follows from (3.50) that

$$\begin{aligned} \bar{Q}W\bar{Q}^* &= \mathbf{1}_T\mathbf{1}_T^\top \otimes \hat{Q}W\hat{Q} \\ &= \mathbf{1}_T\mathbf{1}_T^\top \otimes (Q^{-1} + \sum_{t=1}^T (\hat{E}_3^t)^{-1}) \\ &= \mathbf{1}_T\mathbf{1}_T^\top \otimes ((1 + \sum_{t=1}^T \frac{1}{m_t})Q^{-1})^{-1} \\ &= \mathbf{1}_T\mathbf{1}_T^\top \otimes (1 + \sum_{t=1}^T \frac{1}{m_t})^{-1}Q. \end{aligned}$$

Denote $\bar{m} := (1 + \sum_{t=1}^T \frac{1}{m_t})^{-1}$. Therefore, we have

$$\begin{aligned} (\hat{E}_3 + \hat{E}_4)^{-1} &= \hat{E}_3^{-1} - \hat{E}_3^{-1} \bar{Q} W \bar{Q}^* \hat{E}_3^{-1} \\ &= \left[\begin{bmatrix} \frac{1}{m_1} & & \\ & \ddots & \\ & & \frac{1}{m_T} \end{bmatrix} - \begin{bmatrix} \frac{\bar{m}}{m_1 m_1} & \cdots & \frac{\bar{m}}{m_1 m_T} \\ & \ddots & \\ \frac{\bar{m}}{m_T m_1} & \cdots & \frac{\bar{m}}{m_T m_T} \end{bmatrix} \right] \otimes Q^{-1}. \end{aligned}$$

Hence,

$$y_2 = \text{vec} \left(\left[Q^{-1} \hat{R}_2^1, \dots, Q^{-1} \hat{R}_2^T \right] \left[\begin{bmatrix} \frac{1}{m_1} & & \\ & \ddots & \\ & & \frac{1}{m_T} \end{bmatrix} - \begin{bmatrix} \frac{\bar{m}}{m_1 m_1} & \cdots & \frac{\bar{m}}{m_1 m_T} \\ & \ddots & \\ \frac{\bar{m}}{m_T m_1} & \cdots & \frac{\bar{m}}{m_T m_T} \end{bmatrix} \right] \right). \quad (3.51)$$

Recall that $\hat{R}_2 = R_2 - E_2^* E_1^{-1} R_1 - \frac{1}{m} E_5 R_3$. By the definition of E_1, E_2 , and E_5 in (3.41), we have

$$\hat{R}_2 = \begin{bmatrix} R_2^1 - \frac{\sum_{j=1}^{m_1} (R_{1j}^1) - R_3}{m} \mathbf{1}_{m-1} \\ \vdots \\ R_2^T - \frac{\sum_{j=1}^{m_T} (R_{1j}^T) - R_3}{m} \mathbf{1}_{m-1} \end{bmatrix}.$$

It follows that

$$Q^{-1} \hat{R}_2^t = R_2^t + (\mathbf{1}_{m-1}^\top R_2^t - \mathbf{1}_{m_t}^\top R_1^t + R_3) \mathbf{1}_{m-1}, \quad t = 1, \dots, T.$$

Define $\hat{y}_2^t := R_2^t + (\mathbf{1}_{m-1}^\top R_2^t - \mathbf{1}_{m_t}^\top R_1^t + R_3) \mathbf{1}_{m-1}, t = 1, \dots, T$, and define $\hat{y}_2^a := \sum_{t=1}^T \frac{\bar{m}}{m_t} \hat{y}_2^t$. From (3.51), we have

$$y_2^t = \frac{1}{m_t} (\hat{y}_2^t - \hat{y}_2^a), \quad t = 1, \dots, T.$$

This completes the proof. $\square$

Now, we can summarize the procedure for solving equation $AA^*y = R$ in Algorithm 6.

Algorithm 6 A linear-time complexity solver for the linear system $AA^*y = R$

- 1: **Input:** $R \in \mathbb{R}^M$
 - 2: Step 1. Compute y_2 by (3.43)
 - 3: Step 2. Compute y_1 by (3.44)
 - 4: Step 3. Compute y_3 by (3.45)
 - 5: **Output:** $y = (y_1, y_2, y_3) \in \mathbb{R}^{M_1} \times \mathbb{R}^{M_2} \times \mathbb{R}$
-

Proposition 3.4. *The computational complexity of Algorithm 6 in terms of flops is $7Tm + 3 \sum_{t=1}^T m_t + O(T)$.*

Proof. The complexity of each step in Algorithm 6 can be summarized as follows:

$$\begin{aligned} \text{Step1} &: 6Tm + \sum_{t=1}^T m_t + O(T), \\ \text{Step2} &: Tm + 2 \sum_{t=1}^T m_t + O(T), \\ \text{Step3} &: O(T). \end{aligned}$$

Summing them up, we obtain that the overall computational complexity of Algorithm 6 is $7Tm + 3 \sum_{t=1}^T m_t + O(T)$. $\square$

Remark 3.3. *This complexity analysis can be extended to the OT problem. It is easy to see that the OT problem (1.3) can be reformulated into the following form:*

$$\begin{aligned} \min_x \quad & \langle c, x \rangle + \delta_K(x) \\ \text{s.t.} \quad & Ax = b, \end{aligned} \tag{3.52}$$

where

1. $x = \text{vec}(\pi)$, K is the nonnegative orthant with compatible dimension,

2. $b = (\mathbf{a}^v; [\mathbf{0}_{m_u-1}, I_{m_u-1}]\mathbf{a}^u)$, $c = \text{vec}(\mathcal{D}(\mathcal{P}^u, \mathcal{P}^v))$,

$$3. \ A = \begin{bmatrix} I_{m_v} \otimes \mathbf{1}_{m_u}^\top \\ \mathbf{1}_{m_v}^\top \otimes [\mathbf{0}_{m_u-1}, I_{m_u-1}] \end{bmatrix}.$$

From (Dantzig and Thapa, 2003, Lemma 7.1), we know that A has full row rank.

The solution to the involved linear system

$$AA^*y = AA^* \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$$

can be obtained by

$$\begin{cases} y_1 &= \frac{R_1}{m_u} + \frac{1}{m_v} \left(\frac{m_u-1}{m_u} \mathbf{1}_{m_v}^\top R_1 - \mathbf{1}_{m_u-1}^\top R_2 \right) \mathbf{1}_{m_v}, \\ y_2 &= \frac{R_2}{m_v} + \frac{1}{m_v} \left(\mathbf{1}_{m_u-1}^\top R_2 - \mathbf{1}_{m_v}^\top R_1 \right) \mathbf{1}_{m_u-1}, \end{cases} \quad (3.53)$$

where $R := (R_1, R_2) \in \mathbb{R}^{m_v} \times \mathbb{R}^{(m_u-1)}$. The computational complexity of solving this system by (3.53) is $O(m_u + m_v)$.

We are now ready to present the per-iteration computational complexity of Algorithm 5, which is $O\left(m \sum_{t=1}^T m_t\right)$, as shown in the following lemma. This indicates that the per-iteration cost of the HPR algorithm is comparable to that of the IBP algorithm (Benamou et al., 2015) for solving the WBP.

Lemma 3.2. *The per-iteration computational complexity of Algorithm 5 in terms of flops is $26m \sum_t m_t + O(Tm + \sum_{t=1}^T m_t)$.*

Proof. The computational complexity of k -th ($\forall k \geq 0$) step of Algorithm 5 can be summarized as follows:

Step 1. Since $z^{k+1} := \Pi_K(c - A^*y^k - \tilde{x}^k/\sigma) = \max(0, c - A^*y^k - \tilde{x}^k/\sigma)$, and the matrix A has $2(m-1) \sum_{t=1}^T m_t + T(m-1) + m$ non-zero entries, the complexity for updating z^{k+1} is $7m \sum_{t=1}^T m_t + O(mT)$. Here, we omit $O(Tm)$ since $m_t \geq 1$ for $1 \leq t \leq T$.

Step 3. The complexity of forming $R = \left(b/\sigma - A(x^{k+\frac{1}{2}}/\sigma + z^{k+1} - c)\right)$ is $6m \sum_{t=1}^T m_t + O(Tm)$ due to the structure of A . From Proposition 3.4, we know that the complexity for solving the equation $AA^*y = R$ is $O(Tm + \sum_{t=1}^T m_t)$. Therefore, the complexity of Step 3 is $6m \sum_{t=1}^T m_t + O(Tm + \sum_{t=1}^T m_t)$.

Steps 2, 4, and 5. By some simple calculation, we can see the complexity for these three steps is $13m \sum_{t=1}^T m_t + O(Tm)$.

Summing them up, we know the per-iteration computational complexity of Algorithm 5 in terms of flops is $26m \sum_t m_t + O(Tm + \sum_{t=1}^T m_t)$. $\square$

Now, we can obtain the overall computational complexity of the HPR for solving the WBP.

Theorem 3.2. *Let $\{(y^k, z^k, x^k)\}$ be the sequence generated by Algorithm 5. For any given tolerance $\varepsilon > 0$, HPR needs at most*

$$\frac{1}{\varepsilon} \left(\frac{1+\sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) - 1$$

iterations to return a solution to WBP such that the KKT residual $\|\mathcal{R}(y^{k+1}, z^{k+1}, x^{k+1})\| \leq \varepsilon$, where (x^*, z^*) is the limit point of the sequence $\{(x^k, z^k)\}$. In particular, the overall computational complexity of HPR to achieve this accuracy in terms of flops is

$$O \left(\left(\frac{1+\sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) \frac{m \sum_{t=1}^T m_t}{\varepsilon} \right).$$

Proof. Since $B_2 = I$, it follows from Theorem 3.1 that, for the linear programming (3.38), we have

$$\|\mathcal{R}(y^{k+1}, z^{k+1}, x^{k+1})\| \leq \frac{1}{k+1} \left(\frac{1+\sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) \quad \forall k \geq 0.$$

Therefore, $\|\mathcal{R}(y^{k+1}, z^{k+1}, x^{k+1})\| \leq \varepsilon$ if

$$k \geq \frac{1}{\varepsilon} \left(\frac{1+\sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) - 1.$$

According to Lemma 3.2, the per-iteration computational complexity in terms of the flops of the HPR algorithm for solving the WBP is $O(m \sum_{t=1}^T m_t)$. The overall computational complexity of HPR to achieve this accuracy in terms of flops is

$$O \left(\left(\frac{1+\sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) \frac{m \sum_{t=1}^T m_t}{\varepsilon} \right).$$

This completes the proof. □

Remark 3.4. *Actually, we can reformulate the primal LP model of the WBP (3.37) equivalently as*

$$\begin{aligned} \min_{x \in \mathbb{R}^N, s \in \mathbb{R}^N} \quad & (\langle c, x \rangle + \delta_K(x)) + \delta_\Omega(s) \\ \text{s.t.} \quad & x - s = 0, \end{aligned} \tag{3.54}$$

where δ_Ω is the indicator function of the set $\Omega := \{s \in \mathbb{R}^N \mid As - b = 0\}$. Since the optimization problem (3.54) is in the form of (3.1), the HPR algorithm can be applied to solve it. A key step in the HPR algorithm for solving (3.54) is to compute the projection of a given point $\hat{s} \in \mathbb{R}^N$ onto the set Ω , which is given by

$$\Pi_\Omega(\hat{s}) = \hat{s} - A^*(AA^*)^{-1}(A\hat{s} - b).$$

The projection $\Pi_\Omega(\hat{s})$ can be efficiently computed by applying Algorithm 6. As a consequence, we can also obtain the $O((m \sum_{t=1}^T m_t)/\epsilon)$ computational complexity of the HPR algorithm for solving the WBP in terms of the primal objective function value gap. In this thesis, we prefer to apply the HPR algorithm to solve the dual problem (3.38) partially because it can be more memory efficient.

3.3 Experiments

In this section, we present the numerical performance of the HPR algorithm for solving the WBP with fixed supports on both synthetic and real data sets. We use the 2-Wasserstein distance in all our experiments. We will compare the performance of the HPR algorithm with the fast-ADMM, IBP (Benamou et al., 2015; Schmitzer, 2019), and the commercial software Gurobi. All the numerical experiments in this section are obtained by running MATLAB R2022a on a desktop with Intel(R) Core i7-10700HQ CPU @2.90GHz and 32GB of RAM.

We apply Algorithm 6 to solve the linear system involved in ADMM and call the algorithm fast-ADMM. We adopt the following stopping criterion based on the

relative KKT residual for the HPR algorithm and the fast-ADMM:

$$\text{KKT}_{\text{res}} = \max \left\{ \frac{\|b - Ax\|}{1 + \|b\|}, \frac{\|\min(x, \mathbf{0})\|}{1 + \|x\|}, \frac{\|A^*y + z - c\|}{1 + \|c\| + \|z\|}, \frac{\|z - \Pi_K(z - x)\|}{1 + \|x\| + \|z\|} \right\} \leq 10^{-5}. \quad (3.55)$$

In the implementation, we check the relative KKT residual every 50 steps. For the ADMM algorithm, we will follow the algorithm proposed in (Chen et al., 2021), where the dual stepsize γ can be chosen in $(0, 2)$. Based on our numerical testing, we observe that the fast-ADMM with $\gamma = 1.9$ has better performance. Hence, we set $\gamma = 1.9$ as the default dual stepsize in the fast-ADMM. Also, we consider a hybrid of HPR and fast-ADMM called HPR-hybrid. The specific hybrid strategy is as follows:

$$\begin{cases} \text{Run fast-ADMM,} & \text{if } k \leq 800 \text{ and } \text{KKT}_{\text{res}} \geq 2 \times 10^{-4}, \\ \text{Run HPR,} & \text{otherwise,} \end{cases}$$

where k is the iteration number. In addition, we find that restarting is very useful for improving the performance of HPR. In this paper, we adopt the following restart strategy for HPR:

$$\begin{cases} \text{every 50 iterations,} & \text{if } k \leq 500, \\ \text{KKT}_{\text{res}}^{\text{now}} > \text{KKT}_{\text{res}}^{\text{old}} & \text{or } \text{mod}(k, 500) == 0, \text{ if } k > 500, \end{cases}$$

here $\text{KKT}_{\text{res}}^{\text{old}}$ is the relative KKT residual of the last checking and $\text{KKT}_{\text{res}}^{\text{now}}$ is the relative KKT residual of the current checking. Note that we will check the relative KKT residual every 50 iterations. It is clear that the hybrid strategy does not affect the algorithm's global convergence guarantees. For additional discussion of the restart strategy, see Subsection 5.3.2, which provides further numerical results evaluating the HPR method for general LP.

For IBP, the regularization parameters are chosen from $\{0.01, 0.001\}$ for the synthetic data sets and $\{0.01, 0.001, 0.0005\}$ for real data sets, respectively. If sample distributions share the same ground cost $\{\mathcal{D}(\mathcal{P}^c, \mathcal{P}^t)\}_{t=1}^T$, then we run the IBP im-

plemented in the POT toolbox¹, which is a standard and highly efficient numerical solver for the WBP. Otherwise, we use the Matlab code of the IBP implemented by Yang et al. (2021). For convenience, we call them POT and IBP, respectively. When the regularization parameter is less than 0.01, we will run both the IBP and its stabilized version simultaneously, and will only report the faster results in terms of running time. We adopt the default stopping criteria in the POT and IBP and set $\text{Tol}_{\text{IBP}} = 10^{-6}$.

In this paper, we also compare with the interior point method (IPM) implemented in Gurobi (v9.50 with an academic license). We adopt the default stopping criteria in Gurobi and set the tolerance to be 10^{-6} . We disable the pre-solving phase as well as the cross-over strategy. There are three reasons for choosing the aforementioned settings. First, as observed from our experiments, other methods (such as the primal/dual simplex method) implemented in Gurobi are, in general, not as efficient as the IPM. Second, enabling the pre-solving stage did not improve the performance of the IPM in our numerical tests. Third, the crossover strategy is usually too costly for our experiments, and we do not require a basic solution.

We set the maximum number of iterations as 10000 for all algorithms. The maximum running time (for each run) of all algorithms is set to be **one hour**. For the evaluation of the quality of the solution, we report the KKT_{res} for Gurobi, fast-ADMM, HPR, and HPR-hybrid. To compare with IBP, we will use '**relative obj gap**' and '**relative primal feasibility error**'. Specifically, '**relative obj gap**' stands for the relative objective value gap, which is defined by

$$\text{relative obj gap} = \frac{\left| \sum_{t=1}^T \langle D^t, X^t \rangle - \sum_{t=1}^T \langle D^t, X_g^t \rangle \right|}{\left| \sum_{t=1}^T \langle D^t, X_g^t \rangle \right| + 1}, \quad (3.56)$$

where $(X^1, \dots, X^T)$ is the solution obtained by the algorithm, $(X_g^1, \dots, X_g^T)$ denotes

¹ <https://pythonot.github.io>

the solution obtained by Gurobi; and $D^t = \omega_t \mathcal{D}(\mathcal{P}^c, \mathcal{P}^t)$, $t = 1, \dots, T$. “**relative primal feasibility error**” denotes the value of

$$\max \left\{ \frac{\|b - Ax\|}{1 + \|b\|}, \frac{\|\min(x, \mathbf{0})\|}{1 + \|x\|} \right\}.$$

In order to compare the efficiency of the algorithms, we will report the computational time (in seconds) and the number of iterations.

3.3.1 Experiments on Synthetic Data

In this subsection, we randomly generated a set of discrete probability distributions $\{\mathcal{P}^t\}_{t=1}^T$ with $\mathcal{P}^t = \{(a_i^t, \mathbf{q}_i^t) \in \mathbb{R}_+ \times \mathbb{R}^d : i = 1, \dots, m_t\}$ and $\sum_{i=1}^{m_t} a_i^t = 1$. Specifically, we generate the supports $\{\mathbf{q}_i^t : i = 1, \dots, m_t, t = 1, \dots, T\}$ whose entries are drawn from a Gaussian mixture distribution via the Matlab commands following (Yang et al., 2021):

```
gm_num = 5; gm_mean = [-20; -10; 0; 10; 20];
sigma = zeros(1, 1, gm_num); sigma(1, 1, :) = 5 * ones(gm_num, 1);
gm_weights = rand(gm_num, 1);
distrib = gmdistribution(gm_mean, sigma, gm_weights).
```

The associated weight vector $(a_1^t, \dots, a_{m_t}^t)$ for $\mathcal{P}^t$, $t = 1, \dots, T$ are drawn from the standard uniform distribution on the open interval $(0, 1)$, and then normalize it such that $\sum_{i=1}^{m_t} a_i^t = 1$. Similarly, we also randomly generate the weight vectors $(\omega_1, \dots, \omega_T)$. After generating all $\{\mathcal{P}^t\}_{t=1}^T$, to compute a Wasserstein barycenter $\mathcal{P}^c = \{(a_i^c, \mathbf{q}_i^c) \in \mathbb{R}_+ \times \mathbb{R}^d : i = 1, \dots, m\}$, we first use the k-means² method to select m points from $\{\mathbf{q}_i^t : i = 1, \dots, m_t, t = 1, \dots, T\}$ to be the support points of $\mathcal{P}^c$. For

² In our experiments, we call the Matlab function “kmeans”, which is built-in the statistics and machine learning toolbox.

each $t = 1, \dots, T$, the distance matrix is obtained by

$$\mathcal{D}(\mathcal{P}^c, \mathcal{P}^t)_{ij} = \|q_i^c - q_j^t\|_2^2 \quad (3.57)$$

for $i = 1, \dots, m$ and $j = 1, \dots, m_t$ and is normalized to have $\max_{t=1}^T \|\mathcal{D}(\mathcal{P}^c, \mathcal{P}^t)\|_\infty = 1$. Then we run the previously mentioned methods to solve WBP with fixed supports in (1.5). It should be mentioned that $\mathcal{D}(\mathcal{P}^c, \mathcal{P}^1) = \dots = \mathcal{D}(\mathcal{P}^c, \mathcal{P}^T)$ does not hold for this random data. As a result, POT is not applicable, and we omit the comparison with POT in this subsection. In addition, for convenience, we set $d = 3$ and $m_1 = \dots = m_T = m_t$, and choose different (T, m, m_t) . Then, given each triple (T, m, m_t) , we randomly generate a trial, where each distribution has dense weights and different support points. All results presented are the average of 10 independent trials.

Table 3.1: Numerical results on Gaussian mixture distributions

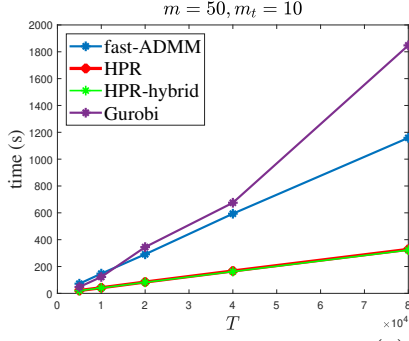
			Gurobi	fast-ADMM	HPR	HPR-hybrid	IBP(0.01)	IBP(0.001)
m	m_t	T	KKT _{res}					
100	100	100	1.73E-07	9.85E-06	9.80E-06	9.77E-06	-	-
100	100	200	1.40E-07	9.92E-06	9.84E-06	9.66E-06	-	-
100	100	400	1.50E-07	9.94E-06	9.67E-06	9.82E-06	-	-
100	100	800	1.72E-07	9.95E-06	9.90E-06	9.88E-06	-	-
200	100	100	2.22E-07	9.89E-06	9.81E-06	9.66E-06	-	-
400	100	100	1.43E-07	9.92E-06	9.78E-06	9.67E-06	-	-
800	100	100	1.23E-07	9.93E-06	9.92E-06	9.71E-06	-	-
100	200	100	2.12E-07	9.92E-06	9.77E-06	9.78E-06	-	-
100	400	100	6.00E-08	9.90E-06	9.57E-06	9.68E-06	-	-
100	800	100	3.16E-08	9.86E-06	9.73E-06	9.55E-06	-	-
m	m_t	T	relative obj gap					
100	100	100	0	4.39E-05	9.31E-05	6.74E-05	3.29E-01	1.60E-02
100	100	200	0	5.79E-05	1.11E-04	8.44E-05	4.11E-01	1.92E-02
100	100	400	0	6.82E-05	1.26E-04	9.64E-05	4.81E-01	2.23E-02
100	100	800	0	8.01E-05	1.30E-04	1.06E-04	5.34E-01	2.40E-02
200	100	100	0	6.07E-05	1.10E-04	9.84E-05	3.61E-01	2.32E-02
400	100	100	0	8.31E-05	1.39E-04	1.29E-04	3.80E-01	3.10E-02
800	100	100	0	1.63E-04	1.94E-04	2.10E-04	4.10E-01	4.06E-02
100	200	100	0	5.81E-05	1.35E-04	1.16E-04	3.77E-01	1.56E-02
100	400	100	0	5.99E-05	1.72E-04	1.65E-04	4.20E-01	1.77E-02
100	800	100	0	7.55E-05	1.66E-04	1.90E-04	4.37E-01	1.75E-02
m	m_t	T	iter					
100	100	100	39	3558	1515	1320	190	3060
100	100	200	54	3978	1615	1340	200	4220
100	100	400	48	4368	1748	1395	210	6120
100	100	800	45	4743	1988	1423	210	7530
200	100	100	49	3270	1713	1363	200	2110
400	100	100	51	3253	1860	1403	200	1490
800	100	100	55	3070	2455	1560	170	1270
100	200	100	36	3473	1565	1303	200	2440
100	400	100	47	3030	1465	1255	200	2400
100	800	100	49	2485	1605	1250	210	2070
m	m_t	T	relative primal feasibility error					
100	100	100	1.76E-09	9.70E-06	9.68E-06	8.60E-06	5.89E-08	5.71E-07
100	100	200	1.70E-09	9.66E-06	9.61E-06	7.98E-06	3.57E-08	6.65E-07
100	100	400	1.07E-09	9.61E-06	9.44E-06	8.43E-06	3.50E-08	6.84E-07
100	100	800	3.88E-09	9.40E-06	8.71E-06	8.54E-06	2.09E-08	1.16E-06
200	100	100	1.54E-09	9.35E-06	9.74E-06	8.83E-06	1.41E-09	4.88E-07
400	100	100	1.57E-09	9.32E-06	9.78E-06	8.38E-06	1.78E-09	3.27E-07
800	100	100	1.57E-09	9.23E-06	9.39E-06	8.81E-06	6.39E-08	2.70E-07
100	200	100	1.52E-09	9.85E-06	8.94E-06	8.27E-06	2.69E-09	5.67E-07
100	400	100	1.37E-09	9.88E-06	9.08E-06	8.03E-06	4.19E-09	5.22E-07
100	800	100	2.62E-12	9.72E-06	9.17E-06	7.86E-06	7.86E-10	4.20E-07
m	m_t	T	time(s)					
100	100	100	7.33	14.66	5.46	5.25	0.99	16.28
100	100	200	23.23	36.70	13.67	12.00	2.20	46.62
100	100	400	40.71	81.53	30.66	25.44	4.59	130.67
100	100	800	70.43	176.82	70.14	52.50	8.71	310.59
200	100	100	20.20	29.62	14.20	11.93	2.17	22.91
400	100	100	44.74	59.64	32.23	24.83	4.20	31.32
800	100	100	202.25	111.45	83.87	54.97	6.97	51.68
100	200	100	12.77	31.11	12.72	11.15	2.05	24.66
100	400	100	34.27	54.68	24.83	22.04	4.16	49.90
100	800	100	224.91	90.35	56.09	44.46	8.68	85.93

A key purpose of the numerical experiments is to demonstrate that the proposed algorithms in this paper can obtain better solutions in terms of solution quality than the entropy regularization-based algorithms, such as the IBP and its stabilized version, and in comparable computational time. Thus, we will focus on the comparison with the IBP. But we want to briefly mention the comparison between the fast-ADMM and the sGS-ADMM (Yang et al., 2021). Based on our testing, in order to obtain solutions with comparable quality, in general, the sGS-ADMM needs about 20% to 30% more iterations than the fast-ADMM. In terms of the per-iteration computational time, the sGS-ADMM needs about 70% more computational time than the fast-ADMM.

The numerical results of the synthetic data are summarized in Table 3.1. Next, we briefly discuss the numerical results. First, the numerical results show that, under reasonable feasibility error, the quality of the solutions obtained by the fast-ADMM, HPR, and HPR-hybrid is better, as implied by the relative objective function value gap. We will further justify this point in the numerical experiments on the real datasets. Second, the IBP algorithm in terms of the per-iteration computational time is extremely fast, but the performance of the IBP is sensitive to the regularization parameter. Third, the per-iteration computational cost of the fast-ADMM is very economical, which partially demonstrates the importance of a linear time complexity procedure in Algorithm 6. Fourth, the HPR algorithm and the HPR-hybrid algorithm are faster than the fast-ADMM. We want to highlight that, due to our extensive numerical testing, we observe that the HPR-hybrid usually outperforms the fast-ADMM and the HPR algorithm. One reason is that the fast-ADMM usually performs quite well, and the HPR can accelerate the performance of the fast-ADMM when the fast-ADMM reaches the bottleneck, since the HPR has a much better worst-case complexity guarantee than the ADMM.

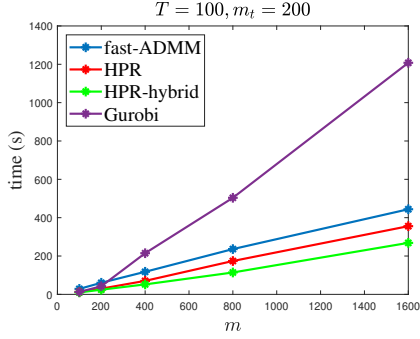
Next, we compare the numerical performance of the fast-ADMM and the HPR

algorithm with Gurobi. When m, m_t , and T are relatively small, Gurobi can solve the problem efficiently. It is a little bit surprising that HPR and HPR-hybrid also have comparable performance in terms of time on small problems. For some large examples like $(m, m_t, T) = (100, 800, 100)$, HPR-hybrid can be 4 times faster than Gurobi. To further compare the performances of Gurobi with fast-ADMM, HPR, and HPR-hybrid, we conduct more experiments on synthetic data. For triple (m, m_t, T) , we fixed two of them and varied the other one. Figure 3.1 shows that fast-ADMM, HPR, and HPR-hybrid always return a similar objective value as Gurobi and have a good feasibility accuracy. For the computational time, fast-ADMM, HPR, and HPR-hybrid increase almost linearly with respect to m, m_t and T , which verifies the complexity result in Lemma 3.2. However, the computational time of Gurobi increases much more rapidly because the complexity of sparse Cholesky decomposition is not linearly related to the dimensionality of variables, and it consumes too much memory for large-scale problems.



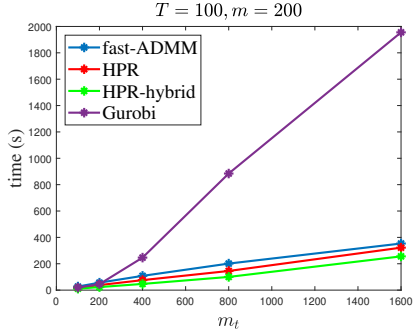
(a) T varies with $m = 50, m_t = 10$

T		relative obj gap				relative primal feasibility error			
		Gurobi	fast ADMM	HPR	HPR-hybrid	Gurobi	fast ADMM	HPR	HPR-hybrid
5000	0		4.9E-06	1.2E-05	1.2E-05	9.7E-10	9.6E-06	8.9E-06	8.9E-06
10000	0		4.3E-06	1.1E-05	1.1E-05	3.6E-09	9.8E-06	8.8E-06	8.8E-06
20000	0		3.8E-06	1.1E-05	1.1E-05	7.2E-09	9.8E-06	9.4E-06	9.4E-06
40000	0		4.8E-06	1.2E-05	1.2E-05	2.4E-09	9.8E-06	9.3E-06	9.3E-06
80000	0		3.6E-06	9.2E-06	9.2E-06	2.1E-08	9.9E-06	9.4E-06	9.4E-06



(b) m varies with $T = 100, m_t = 200$

m		relative obj gap				relative primal feasibility error			
		Gurobi	fast ADMM	HPR	HPR-hybrid	Gurobi	fast ADMM	HPR	HPR-hybrid
100	0		5.6E-05	1.4E-04	1.2E-04	2.2E-09	9.6E-06	9.1E-06	8.3E-06
200	0		1.0E-04	1.8E-04	1.6E-04	1.7E-09	9.5E-06	9.6E-06	8.6E-06
400	0		1.4E-04	2.0E-04	1.9E-04	1.8E-09	9.3E-06	9.6E-06	8.8E-06
800	0		2.2E-04	2.6E-04	3.1E-04	2.2E-09	9.0E-06	9.8E-06	9.0E-06
1600	0		3.5E-04	3.9E-04	5.3E-04	2.3E-09	8.7E-06	9.6E-06	8.9E-06



(c) m_t varies with $m = 200, T = 100$

m_t		relative obj gap				relative primal feasibility error			
		Gurobi	fast ADMM	HPR	HPR-hybrid	Gurobi	fast ADMM	HPR	HPR-hybrid
100	0		6.5E-05	1.0E-04	1.0E-04	2.0E-09	9.4E-06	9.6E-06	8.6E-06
200	0		1.0E-04	1.9E-04	1.7E-04	1.3E-09	9.6E-06	9.4E-06	8.6E-06
400	0		1.5E-04	2.6E-04	2.5E-04	1.6E-09	9.3E-06	9.5E-06	8.7E-06
800	0		1.6E-04	2.8E-04	3.0E-04	2.7E-11	9.5E-06	9.2E-06	8.7E-06
1600	0		1.7E-04	3.2E-04	2.5E-04	5.4E-13	9.1E-06	9.5E-06	9.0E-06

Figure 3.1: Comparisons between Gurobi, fast-ADMM, HPR, and HPR-hybrid

Numerical Comparison of the Constants in the Complexity Bounds. Now, we will compare the constants of the complexity bounds between the HPR algorithm and the entropy regularized algorithms, such as the IBP. The constant of the complexity bounds of the IBP depends on $\max_{t=1}^T \|\mathcal{D}(\mathcal{P}^c, \mathcal{P}^t)\|_\infty$, which is one in our synthetic data settings. On the other hand, the constant for the HPR depends on

the distance between the initial point and the solution, as shown in Theorem 3.2. Since we choose zeros as the initial point of the HPR algorithm, we only need to compute $\|x^*\|$ and $\|z^*\|$. We test it on the synthetic data with two different settings. For $(T, m, m_t) = (100, 100, 100)$, we get $\|x^*\| = 1.10$ and $\|z^*\| = 3.52$. For $(T, m, m_t) = (100, 100, 800)$, we get $\|x^*\| = 0.98$ and $\|z^*\| = 0.36$.

3.3.2 Experiments on Real Data

To further compare the HPR and HPR-hybrid to other methods, we conduct experiments on some Real Data. Our experiments include the MNIST data set (LeCun et al., 1998), the Coil20 data set (Nene et al., 1996), and the Yale Face B data set (Georghiades et al., 2001). For the MNIST data set, we randomly select 50 images for each digit $(0, \dots, 9)$ and resize each image to 56×56 . For the Coil20 data set, we select 3 representative objects: Car, Duck, and Pig, where each object has 10 images. We resize each image to 64×64 . For the Yale Face B data set, we include two human subjects: YaleB01 and YaleB02. We randomly select 5 images for each human subject and resize them to 96×84 . A summary of each data set is shown in Table 3.2. For all data sets, we normalize the resulting image so that all pixel values add up to 1. We generate the distance matrix similarly to (3.57). At last, we set the weight vector $(\omega_1, \omega_2, \dots, \omega_T)$ such that $\omega_t = 1/T$ for all $t = 1, \dots, T$.

Table 3.2: Summary of real datasets

Dataset	m	m_t	T
MNIST	3136	3136	50
Car	4096	4096	10
Duck	4096	4096	10
Pig	4096	4096	10
YaleB01	8064	8064	5
YaleB02	8064	8064	5

Since Gurobi is out of memory for this experiment, we do not compare with Gurobi in this subsection and do not report the relative objective gap. Following (Yang et al., 2021), we visualize the MNIST results in Figure 3.2, where the Wasserstein barycenters are obtained by different methods for the 50s and 100s, respectively. One can see that HPR and HPR-hybrid can provide a clear “smooth” barycenter just like POT with a regularization parameter $\{0.001\}$ within a fixed time.

For the Coil20 and Yale Face B data sets, we first apply POT to get a barycenter as a benchmark. We find that POT with a regularization parameter $\{0.0005\}$ gives the best possible result. Although we try to use the smaller parameter, IBP implemented in POT encounters the numerical error even with the stabilized version. In this experiment, we want to know how many iterations of fast-ADMM, HPR, and HPR-hybrid are needed to reach the quality of the best solution returned by POT. The result of the Coil20 data set and Yale Face B data set is presented in Figure 3.3 and Figure 3.4, respectively. We list the computational time of different methods on these data sets in Table 3.3. Figures 3.3 and 3.4 show that the quality of solutions produced by fast-ADMM, HPR, and HPR-hybrid is better than that of POT within 300 iterations. In particular, HPR-hybrid is faster and more stable than ADMM and HPR. HPR-hybrid can return a better result than POT in the 100th iteration, whose computational time is comparable to the time of POT from Table 3.3. Hence, we recommend using HPR-hybrid for computing the Wasserstein barycenter with high accuracy requirements in practice.

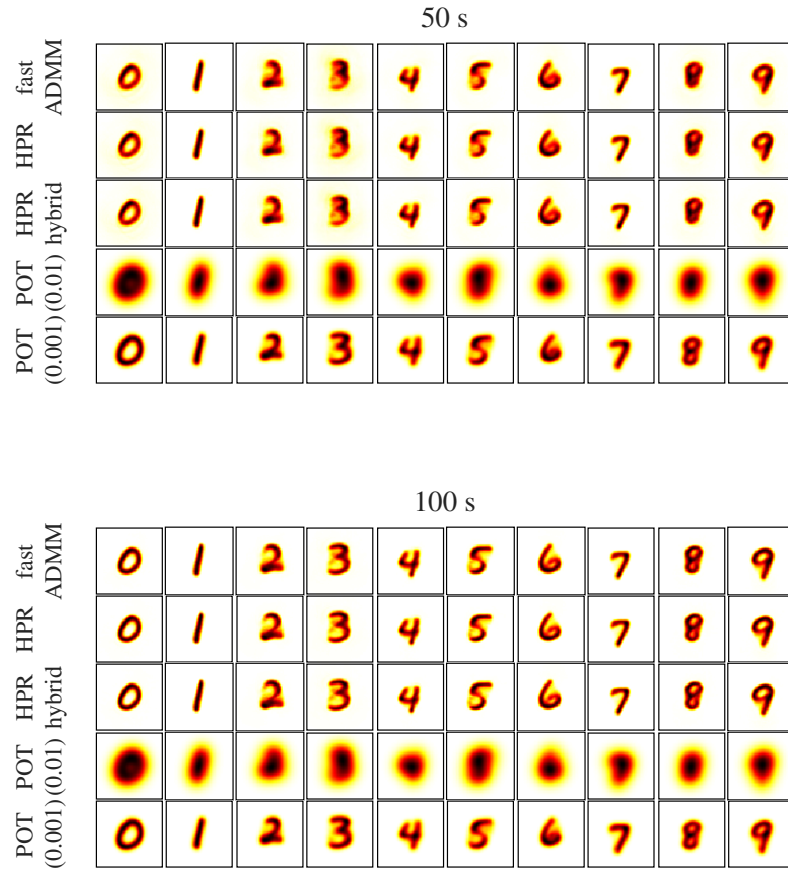


Figure 3.2: Barycenters obtained by running different methods for 50s and 100s, respectively.

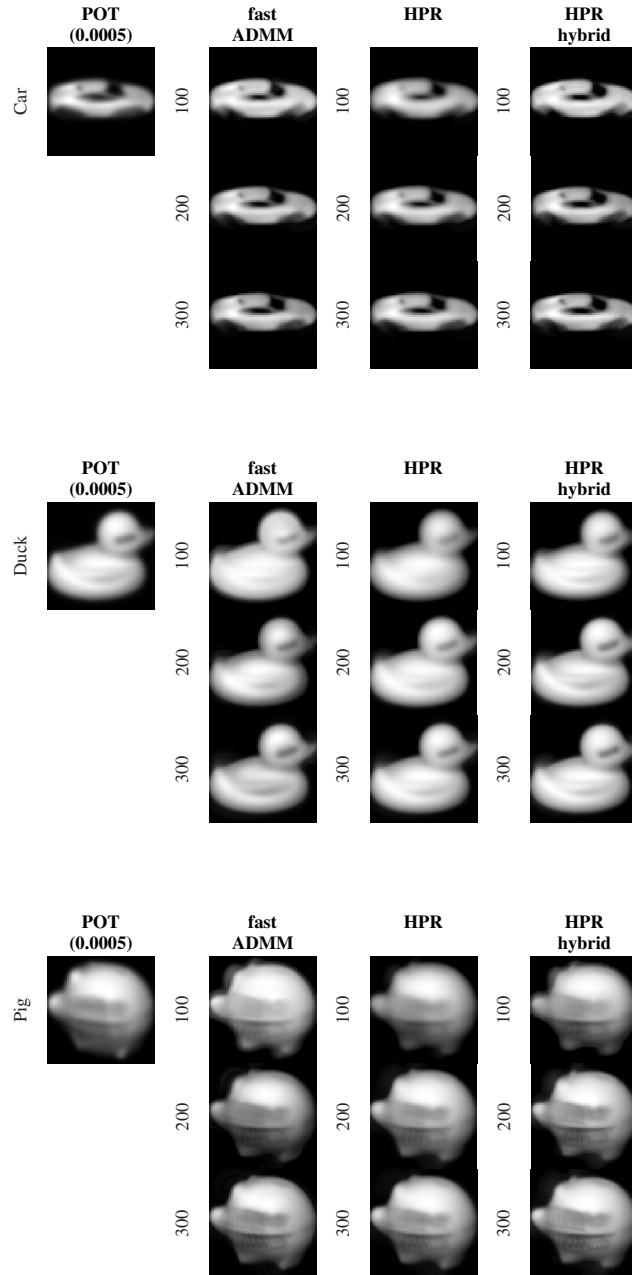


Figure 3.3: Barycenters obtained by running different methods on the Coil20 data set for the 100th, 200th, and 300th iterations.

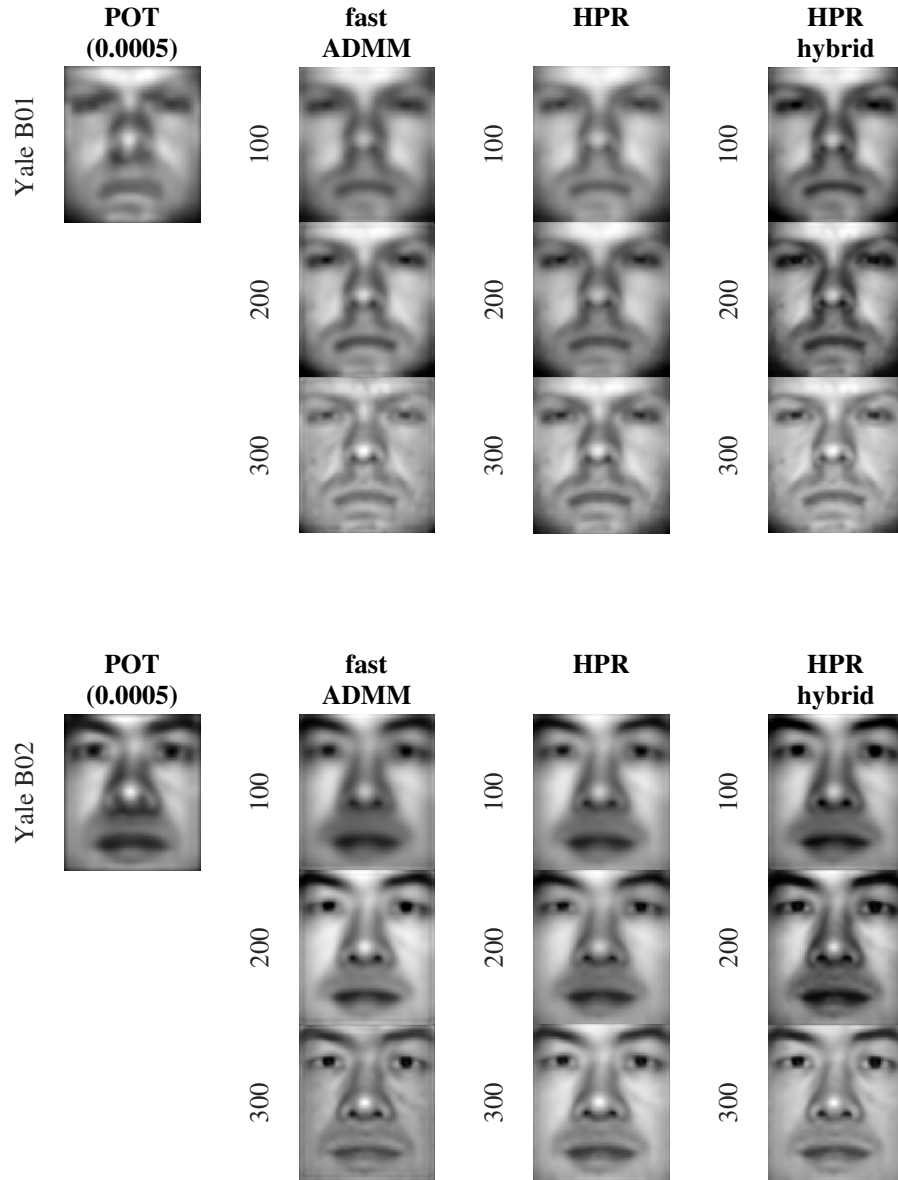


Figure 3.4: Barycenters obtained by running different methods on the Yale Face B data set for the 100th, 200th, and 300th iterations.

Table 3.3: Computation times (in seconds) of different methods for computing the Wasserstein barycenter on the Coil20 and Yale B face datasets.

	POT(0.0005)	fast-ADMM			HPR			HPR-hybrid		
iter	-	100	200	300	100	200	300	100	200	300
Car	44.59	27.61	55.22	82.82	25.60	51.20	76.81	25.79	51.58	77.38
Duck	48.89	63.51	127.02	190.52	60.68	121.36	182.03	62.13	124.25	186.38
Pig	33.25	90.92	181.83	272.75	87.33	174.66	262.00	87.90	175.81	263.71
YaleB01	164.62	58.30	116.60	174.90	55.76	111.51	167.27	56.80	113.60	170.41
YaleB02	153.86	178.52	357.04	535.57	176.35	352.70	529.06	177.34	354.68	532.03

From the numerical results reported in the previous subsections, one can see that HPR and HPR-hybrid outperform the powerful commercial solver Gurobi in terms of the computational time for solving large-scale LPs arising from WBPs. Moreover, to get a high-quality solution, one needs to use a small regularization parameter for IBP-type methods, which will easily suffer from numerical instability issues. Compared with IBP-type methods, HPR-hybrid is more stable and can return a good solution in a comparable time. Hence, for computing a high-quality Wasserstein barycenter, we recommend applying HPR-hybrid to get a high-quality solution without the need to implement sophisticated stabilization techniques as in the case of IBP.

Chapter 4

An Accelerated Preconditioned (Semi-proximal) ADMM Method for OT Problems

In this chapter, we develop an accelerated dPPA framework for monotone inclusion problems. Then we establish the equivalence between pADMM and dPPA. Leveraging this connection, we design an accelerated pADMM algorithm with provable convergence rates. We further apply this framework to develop the HOT algorithm—a Halpern accelerating method for solving a reduced formulation of the OT problem. We also propose a transport plan recovery procedure that overcomes the limitations of reduced models. Numerical results show that HOT outperforms existing state-of-the-art algorithms on large-scale OT problems.

4.1 An Accelerated Degenerate Proximal Point Algorithm

Recall the following monotone inclusion problem:

$$\text{find } w \in \mathbb{W} \text{ such that } 0 \in \mathcal{T}w, \quad (4.1)$$

where $\mathcal{T}$ is a maximal monotone operator from $\mathbb{W}$ into itself. Let $\mathcal{M}$ be an admissible preconditioner according to Definition 2.1 for the maximal monotone operator $\mathcal{T}$.

The dPPA for solving the inclusion problem (4.1) is expressed as follows:

$$w^0 \in \mathbb{W}, \quad w^{k+1} = (1 - \rho_k)w^k + \rho_k \bar{w}^k, \quad \bar{w}^k = \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k = (\mathcal{M} + \mathcal{T})^{-1} \mathcal{M} w^k, \quad (4.2)$$

where $\{\rho_k\}$ is a sequence in $[0, 2]$. In this section, we consider the acceleration of the dPPA with a fixed relaxation parameter $\rho \in (0, 2]$. Specifically, the dPPA in (4.2) can be reformulated as

$$w^0 \in \mathbb{W}, \quad w^{k+1} = \mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} w^k, \quad (4.3)$$

where $\mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} = (1 - \rho)\mathcal{I} + \rho\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$ is $\mathcal{M}$ -nonexpansive for $\rho \in (0, 2]$ according to Proposition 2.2. Leveraging the $\mathcal{M}$ -nonexpansiveness of $\mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}}$, we propose the following accelerated dPPA presented in Algorithm 7.

Algorithm 7 An accelerated dPPA for solving the inclusion problem (4.1)

- 1: **Input:** Let $\hat{w}^0 = w^0 \in \mathbb{W}$, $\alpha \geq 2$, and $\rho \in (0, 2]$
 - 2: **for** $k = 0, 1, \dots$ **do**
 - 3: Step 1. $\bar{w}^k = \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k$
 - 4: Step 2. $\hat{w}^{k+1} = \mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} w^k = (1 - \rho)w^k + \rho\bar{w}^k$
 - 5: Step 3. $w^{k+1} = w^k + \frac{\alpha}{2(k+\alpha)}(\hat{w}^{k+1} - w^k) + \frac{k}{k+\alpha}(\hat{w}^{k+1} - \hat{w}^k)$
-

Remark 4.1. *The proposed accelerated dPPA can be regarded as applying either Halpern's iteration (Lieder, 2021) or the fast KM iteration (Boř and Nguyen, 2023) to the dPPA in (4.3), depending on the choice of α . On the one hand, when $\alpha = 2$, Step 3 in Algorithm 7 at the k -th iteration is*

$$w^{k+1} = w^k + \frac{1}{k+2}(\hat{w}^{k+1} - w^k) + \frac{k}{k+2}(\hat{w}^{k+1} - \hat{w}^k).$$

Multiplying both sides of the above relation by $(k+2)$ and rearranging it, we have

$$(k+2)w^{k+1} - (k+1)\hat{w}^{k+1} = (k+1)w^k - k\hat{w}^k. \quad (4.4)$$

Define the sequence $\{s^k\}$ as follows:

$$s^k := (k+1)w^k - k\hat{w}^k \quad \forall k \geq 0.$$

From (4.4), we can obtain $s^{k+1} = s^0$ for all $k \geq 0$. This implies

$$w^{k+1} = \frac{1}{k+2}w^0 + \frac{k+1}{k+2}\hat{w}^{k+1} \quad \forall k \geq 0, \quad (4.5)$$

which is exactly the Halpern iteration (Halpern, 1967; Lieder, 2021) applied to the dPPA in (4.3). On the other hand, when $\alpha > 2$, the proposed accelerated dPPA is equivalent to the fast KM iteration applied to (4.3) with a unit stepsize, as described in formula (12) of (Boţ and Nguyen, 2023).

Let $\mathcal{M} = \mathcal{C}\mathcal{C}^*$ be a decomposition of $\mathcal{M}$ according to Proposition 2.1 with $\mathcal{C} : \mathbb{U} \rightarrow \mathbb{W}$. To discuss the global convergence and convergence rate of the accelerated dPPA in Algorithm 7, we introduce two shadow sequences $\{u^k\}$ and $\{\bar{u}^k\}$ defined as follows:

$$u^k := \mathcal{C}^*w^k \text{ and } \bar{u}^k := \mathcal{C}^*\bar{w}^k \quad \forall k \geq 0, \quad (4.6)$$

where the sequences $\{w^k\}$ and $\{\bar{w}^k\}$ are generated by Algorithm 7. This leads to the following identity:

$$u^{k+1} = u^k + \frac{\alpha}{2(k+\alpha)}(\mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho}u^k - u^k) + \frac{k}{k+\alpha}(\mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho}u^k - \mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho}u^{k-1}) \quad \forall k \geq 1, \quad (4.7)$$

where $\mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho} = (1-\rho)\mathcal{I} + \rho\mathcal{J}_{\mathcal{C}*\triangleright\mathcal{T}}$. Based on the shadow sequences in (4.6) and the equivalence outlined in Remark 4.1, we establish the global convergence of the proposed accelerated dPPA in the following theorem.

Theorem 4.1. *Let $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ with $\mathcal{T}^{-1}(0) \neq \emptyset$ be a maximal monotone operator, and let $\mathcal{M}$ be an admissible preconditioner such that $(\mathcal{M} + \mathcal{T})^{-1}$ is continuous. Suppose that $\mathcal{M} = \mathcal{C}\mathcal{C}^*$ is a decomposition of $\mathcal{M}$ according to Proposition 2.1 with $\mathcal{C} : \mathbb{U} \rightarrow \mathbb{W}$. The following conclusions hold for the sequences $\{\bar{w}^k\}$, $\{\hat{w}^k\}$, and $\{w^k\}$ generated by the accelerated dPPA in Algorithm 7:*

(a) If $\alpha = 2$, then the sequence $\{\bar{w}^k\}$ converges to a fixed point $w^* = (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(\mathcal{C}^*w^0)$ in $\mathcal{T}^{-1}(0)$; moreover, if $\rho \in (0, 2)$, then the sequences $\{w^k\}$ and $\{\hat{w}^k\}$ also converge to w^* ;

(b) If $\alpha > 2$, then the sequence $\{\bar{w}^k\}$ converges to a fixed-point in $\mathcal{T}^{-1}(0)$.

Proof. We first establish the statement in part (a), where $\alpha = 2$. According to the connection between the accelerated dPPA and the Halpern iteration outlined in (4.5), and the relationship between $\{u^k\}$ and $\{w^k\}$ in (4.6), we can obtain that

$$u^{k+1} = \frac{1}{k+2}u^0 + \frac{k+1}{k+2}\mathcal{F}_{\mathcal{C}^*\triangleright\mathcal{T}, \rho}u^k \quad \forall k \geq 0. \quad (4.8)$$

Note that $\mathcal{F}_{\mathcal{C}^*\triangleright\mathcal{T}, \rho}$ is nonexpansive for any $\rho \in (0, 2]$ by Proposition 2.3. It follows from the global convergence of the Halpern iteration in Theorem 2.2 that

$$u^k \rightarrow \Pi_{\text{Fix } \mathcal{J}_{\mathcal{C}^*\triangleright\mathcal{T}}}(u^0). \quad (4.9)$$

By utilizing Proposition 2.3, we have

$$\mathcal{C}^*\mathcal{T}^{-1}(0) = \mathcal{C}^* \text{Fix } \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} = \text{Fix } \mathcal{J}_{\mathcal{C}^*\triangleright\mathcal{T}}.$$

Let $w^* = (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0)$. Consequently, by the relationship between $\{u^k\}$ and $\{w^k\}$ in (4.6), and (4.9), we can obtain

$$\bar{w}^k = (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\mathcal{C}^*w^k = (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}u^k \rightarrow (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0) = w^*, \quad (4.10)$$

where the continuity of $(\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}$ is derived from the composition of a continuous function $(\mathcal{M} + \mathcal{T})^{-1}$ and a linear operator $\mathcal{C}$. Hence, $\{\bar{w}^k\}$ converges to w^* .

Furthermore, since $\mathcal{J}_{\mathcal{C}^*\triangleright\mathcal{T}}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0) = \Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0)$, we have

$$\begin{aligned} \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^* &= (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\mathcal{C}^*(\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0) \\ &= (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\mathcal{J}_{\mathcal{C}^*\triangleright\mathcal{T}}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0) \\ &= (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\Pi_{\mathcal{C}^*\mathcal{T}^{-1}(0)}(u^0) = w^*, \end{aligned}$$

which implies that $w^* \in \text{Fix } \mathcal{J}_T^M = \mathcal{T}^{-1}(0)$. It remains to show that $\{w^k\}$ and $\{\hat{w}^k\}$ also converge to w^* for $\rho \in (0, 2)$. According to (4.5), we have

$$\begin{aligned} & \|w^{k+1} - w^*\| \\ = & \left\| \frac{1}{k+2}(w^0 - w^*) + \frac{k+1}{k+2}((1-\rho)(w^k - w^*) + \rho(\bar{w}^k - w^*)) \right\| \\ \leq & \frac{k+1}{k+2}|1-\rho|\|w^k - w^*\| + \frac{1}{k+2}\|w^0 - w^*\| + \frac{k+1}{k+2}\rho\|\bar{w}^k - w^*\|. \end{aligned} \quad (4.11)$$

For any $k \geq 0$, define $q_k := \frac{k+1}{k+2}|1-\rho|$ and $\beta_k := \frac{1}{k+2}\|w^0 - w^*\| + \frac{k+1}{k+2}\rho\|\bar{w}^k - w^*\|$. It is evident that $0 \leq q_k < 1$, $\beta_k \geq 0$ for any $k \geq 0$, and

$$\sum_{k=0}^{\infty} (1 - q_k) = \sum_{k=0}^{\infty} \left(1 - \frac{k+1}{k+2}|1-\rho| \right) = \sum_{k=0}^{\infty} \left(\frac{1}{k+2} + \frac{k+1}{k+2}(1 - |1-\rho|) \right) = +\infty. \quad (4.12)$$

Furthermore, due to the convergence of $\{\bar{w}^k\}$, we can deduce

$$\frac{\beta_k}{1 - q_k} \rightarrow 0, \quad \forall \rho \in (0, 2). \quad (4.13)$$

It follows from (4.12), (4.13), and (Polyak, 1987, Lemma 3 in subsection 2.2.1) that

$$\|w^{k+1} - w^*\| \rightarrow 0,$$

which in accordance with (4.5) implies

$$\|\hat{w}^k - w^*\| \rightarrow 0.$$

Hence, $\{w^k\}$ and $\{\hat{w}^k\}$ also converge to w^* for $\rho \in (0, 2)$.

Now, we shall prove part (b) where $\alpha > 2$. According to the identity in (4.7), the nonexpansiveness of $\mathcal{F}_{\mathcal{C}_{*\triangleright\mathcal{T}}, \rho}$, and the convergence result of the fast KM in Theorem 2.4, we can obtain that $\{u^k\}$ converges to an element in $\text{Fix } \mathcal{J}_{\mathcal{C}_{*\triangleright\mathcal{T}}}$. Similar to (4.10), we can deduce that $\{\bar{w}^k\}$ converges to a point in $\mathcal{T}^{-1}(0)$. $\square$

Remark 4.2. *The acceleration step (Step 3) in Algorithm 7 can be replaced by other acceleration techniques (Tran-Dinh, 2024). Following a similar approach to the proof of Theorem 4.1, we can obtain the global convergence of the sequence $\{\bar{w}^k\}$ if the shadow sequence $\{u^k\}$ converges.*

The following proposition provides convergence rates for the accelerated dPPA.

Proposition 4.1. *Let $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ with $\mathcal{T}^{-1}(0) \neq \emptyset$ be a maximal monotone operator and let $\mathcal{M}$ be an admissible preconditioner with closed range. The sequences $\{w^k\}$ and $\{\hat{w}^k\}$ generated by Algorithm 7 satisfy the following:*

(a) *if $\alpha = 2$, then*

$$\|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} \leq \frac{2\|w^0 - w^*\|_{\mathcal{M}}}{k+1} \quad \forall k \geq 0 \text{ and } w^* \in \mathcal{T}^{-1}(0); \quad (4.14)$$

(b) *if $\alpha > 2$, then*

$$\|w^{k+1} - w^k\|_{\mathcal{M}} = o\left(\frac{1}{k+1}\right) \text{ and } \|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} = o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty. \quad (4.15)$$

Proof. Let $\mathcal{M} = \mathcal{C}\mathcal{C}^*$ be a decomposition of $\mathcal{M}$ according to Proposition 2.1 with $\mathcal{C} : \mathbb{U} \rightarrow \mathbb{W}$. If $\alpha = 2$, then the shadow sequence $\{u^k\}$ satisfying (4.8) is exactly the Halpern iteration. By Proposition 2.3, we know that $\mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho}$ is nonexpansive for $\rho \in (0, 2]$. It follows from Theorem 2.3 that

$$\|u^k - \mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho} u^k\| \leq \frac{2\|u^0 - u^*\|}{k+1} \quad \forall k \geq 0 \text{ and } u^* \in \text{Fix } \mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho}. \quad (4.16)$$

Due to Proposition 2.3, we also have $\text{Fix } \mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho} = \mathcal{C}^*\mathcal{T}^{-1}(0)$. Thus, for any $u^* \in \text{Fix } \mathcal{F}_{\mathcal{C}*\triangleright\mathcal{T},\rho}$, there exists a point $w^* = (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}u^* \in \mathcal{T}^{-1}(0)$ such that $\mathcal{C}^*w^* = u^*$. Hence, (4.16) can be rewritten as:

$$\|\mathcal{C}^*w^k - \mathcal{C}^*\mathcal{F}_{\mathcal{T},\rho}^{\mathcal{M}} w^k\| \leq \frac{2\|\mathcal{C}^*w^0 - \mathcal{C}^*w^*\|}{k+1} \quad \forall k \geq 0 \text{ and } w^* \in \mathcal{T}^{-1}(0),$$

which implies

$$\|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} = \|w^k - \mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}} w^k\|_{\mathcal{M}} \leq \frac{2\|w^0 - w^*\|_{\mathcal{M}}}{k+1} \quad \forall k \geq 0 \text{ and } w^* \in \mathcal{T}^{-1}(0).$$

If $\alpha > 2$, then the shadow sequence $\{u^k\}$ satisfying (4.7) is exactly the fast KM iteration. It follows from Theorem 2.5 that

$$\|u^{k+1} - u^k\| = o\left(\frac{1}{k+1}\right) \quad \text{and} \quad \|u^k - \mathcal{F}_{\mathcal{C} \triangleright \mathcal{T}, \rho} u^k\| = o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty.$$

Similar to the previous case with $\alpha = 2$, we can conclude that

$$\|w^{k+1} - w^k\|_{\mathcal{M}} = o\left(\frac{1}{k+1}\right) \quad \text{and} \quad \|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} = o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty,$$

which completes the proof. $\square$

Remark 4.3. *The convergence rates discussed in Proposition 4.1 are essentially inherited from the results of the Halpern iteration (Lieder, 2021) and the fast KM iteration (Boţ and Nguyen, 2023) applied to the nonexpansive operator $\mathcal{F}_{\mathcal{C} \triangleright \mathcal{T}, \rho}$ defined in Proposition 2.3. Moreover, without acceleration, similar to (Brézis and Lions, 1978, Proposition 8), we can obtain that the dPPA with $\rho = 1$ has an $O(1/\sqrt{k})$ convergence rate with respect to $\|w^k - w^{k-1}\|_{\mathcal{M}} \forall k \geq 1$.*

The following corollary demonstrates that even when the proximal term $\mathcal{M}$ is positive semidefinite, the accelerated dPPA can achieve an $O(1/k)$ convergence rate in terms of the operator residual under the norm.

Corollary 4.1. *Let $\mathcal{T} : \mathbb{W} \rightarrow 2^{\mathbb{W}}$ with $\mathcal{T}^{-1}(0) \neq \emptyset$ be a maximal monotone operator and let $\mathcal{M}$ be an admissible preconditioner with closed range such that $(\mathcal{M} + \mathcal{T})^{-1}$ is L -Lipschitz. Suppose that $\mathcal{M} = \mathcal{C}\mathcal{C}^*$ is a decomposition of $\mathcal{M}$ according to Proposition 2.1 with $\mathcal{C} : \mathbb{U} \rightarrow \mathbb{W}$. Choose $\alpha = 2$ and $\rho = 1$. Then the sequences $\{w^k\}$ and $\{\bar{w}^k\}$ generated by Algorithm 7 satisfy*

$$\|w^k - \bar{w}^k\| \leq \frac{1}{k+1} \|w^0 - w^*\| + \frac{(5k+1)L\|\mathcal{C}\|}{(k+1)^2} \|w^0 - w^*\|_{\mathcal{M}} \quad \forall k \geq 0 \text{ and } w^* \in \mathcal{T}^{-1}(0).$$

Proof. Set $\alpha = 2$ and $\rho = 1$. Then according to (4.5), we have

$$w^{k+1} = \frac{1}{k+2}w^0 + \frac{k+1}{k+2}\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^k \quad \forall k \geq 0. \quad (4.17)$$

Since $(\mathcal{M} + \mathcal{T})^{-1}$ is L -Lipschitz, we have for all $w', w'' \in \mathbb{W}$,

$$\|\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w' - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w''\| = \|(\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\mathcal{C}^*w' - (\mathcal{M} + \mathcal{T})^{-1}\mathcal{C}\mathcal{C}^*w''\| \leq L\|\mathcal{C}\| \|w' - w''\|_{\mathcal{M}},$$

which, together with (4.17), yields that for any $w^* \in \mathcal{T}^{-1}(0)$ and $k \geq 0$,

$$\begin{aligned} & \|w^{k+1} - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^{k+1}\| \\ &= \left\| \frac{1}{k+2}[(w^0 - w^*) + (w^* - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^{k+1})] + \frac{k+1}{k+2}(\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^k - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^{k+1}) \right\| \\ &\leq \frac{1}{k+2}(\|w^0 - w^*\| + \|w^* - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^{k+1}\|) + \frac{k+1}{k+2}\|\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^k - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^{k+1}\| \\ &\leq \frac{1}{k+2}(\|w^0 - w^*\| + L\|\mathcal{C}\|\|w^* - w^{k+1}\|_{\mathcal{M}}) + \frac{k+1}{k+2}L\|\mathcal{C}\|\|w^k - w^{k+1}\|_{\mathcal{M}}. \end{aligned} \quad (4.18)$$

Next, we shall estimate $\|w^k - w^{k+1}\|_{\mathcal{M}}$. Indeed, from (4.17), we have

$$\begin{aligned} & \|w^k - w^{k+1}\|_{\mathcal{M}} \\ &= \left\| \frac{1}{k+2}[(w^0 - w^*) + (w^* - w^k)] + \frac{k+1}{k+2}(\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^k - w^k) \right\|_{\mathcal{M}} \\ &\leq \frac{1}{k+2}(\|w^0 - w^*\|_{\mathcal{M}} + \|w^* - w^k\|_{\mathcal{M}}) + \frac{k+1}{k+2}\|\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^k - w^k\|_{\mathcal{M}}. \end{aligned} \quad (4.19)$$

Now, we claim that $\|w^k - w^*\|_{\mathcal{M}} \leq \|w^0 - w^*\|_{\mathcal{M}}$ for any $k \geq 0$. Assume $\|w^k - w^*\|_{\mathcal{M}} \leq \|w^0 - w^*\|_{\mathcal{M}}$ for some $k \geq 0$ (this is true for $k = 0$ trivially). Since $\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}$ is $\mathcal{M}$ -nonexpansive by Proposition 2.2, we have

$$\begin{aligned} \|w^{k+1} - w^*\|_{\mathcal{M}} &= \left\| \frac{1}{k+2}(w^0 - w^*) + \frac{k+1}{k+2}(\mathcal{J}_{\mathcal{T}}^{\mathcal{M}}w^k - w^*) \right\|_{\mathcal{M}} \\ &\leq \frac{1}{k+2}\|w^0 - w^*\|_{\mathcal{M}} + \frac{k+1}{k+2}\|w^k - w^*\|_{\mathcal{M}} \\ &\leq \|w^0 - w^*\|_{\mathcal{M}}. \end{aligned}$$

In particular, this implies that $\|w^1 - w^*\|_{\mathcal{M}} \leq \|w^0 - w^*\|_{\mathcal{M}}$. Thus, we derive from the induction that

$$\|w^k - w^*\|_{\mathcal{M}} \leq \|w^0 - w^*\|_{\mathcal{M}} \quad \forall k \geq 0. \quad (4.20)$$

Thus, from (4.18), (4.19), and (4.20), we can deduce that for any $k \geq 0$,

$$\begin{aligned} \|w^{k+1} - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^{k+1}\| &\leq \frac{1}{k+2} (\|w^0 - w^*\| + L\|\mathcal{C}\|\|w^* - w^0\|_{\mathcal{M}}) \\ &\quad + \frac{k+1}{k+2} L\|\mathcal{C}\| \left(\frac{2}{k+2} \|w^0 - w^*\|_{\mathcal{M}} + \frac{k+1}{k+2} \|\mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k - w^k\|_{\mathcal{M}} \right). \end{aligned} \quad (4.21)$$

Also, from Proposition 4.1, we have for $\rho = 1$,

$$\|w^k - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k\|_{\mathcal{M}} \leq \frac{2\|w^0 - w^*\|_{\mathcal{M}}}{k+1} \quad \forall k \geq 0 \text{ and } w^* \in \mathcal{T}^{-1}(0),$$

which together with (4.21) yields that for any $k \geq 0$,

$$\|w^{k+1} - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^{k+1}\| \leq \frac{1}{k+2} \|w^0 - w^*\| + \frac{5k+6}{(k+2)^2} L\|\mathcal{C}\|\|w^0 - w^*\|_{\mathcal{M}}.$$

Note that for $k = 0$, we also have

$$\begin{aligned} \|w^0 - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^0\| &\leq \|w^0 - w^*\| + \|\mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^0 - w^*\| \\ &\leq \|w^0 - w^*\| + L\|\mathcal{C}\|\|w^0 - w^*\|_{\mathcal{M}}. \end{aligned}$$

Hence, for any $k \geq 0$,

$$\|w^k - \mathcal{J}_{\mathcal{T}}^{\mathcal{M}} w^k\| \leq \frac{1}{k+1} \|w^0 - w^*\| + \frac{5k+1}{(k+1)^2} L\|\mathcal{C}\|\|w^0 - w^*\|_{\mathcal{M}},$$

which completes the proof. $\square$

4.2 Acceleration of the Preconditioned ADMM

In this section, we first establish the equivalence between the pADMM and the dPPA. Subsequently, we introduce an accelerated pADMM based on the accelerated dPPA. Finally, we present the global convergence of the accelerated pADMM and discuss its convergence rate.

4.2.1 The Equivalence Between the pADMM and the dPPA

Recall the linearly constrained COP (4.22) given in Chapter 2:

$$\begin{aligned} \min_{y \in \mathbb{Y}, z \in \mathbb{Z}} \quad & f_1(y) + f_2(z) \\ \text{subject to} \quad & B_1 y + B_2 z = c, \end{aligned} \quad (4.22)$$

where $f_1 : \mathbb{Y} \rightarrow (-\infty, +\infty]$ and $f_2 : \mathbb{Z} \rightarrow (-\infty, +\infty]$ are two proper closed convex functions, $B_1 : \mathbb{Y} \rightarrow \mathbb{X}$ and $B_2 : \mathbb{Z} \rightarrow \mathbb{X}$ 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 (4.22) is defined by, for any $(y, z, x) \in \mathbb{W} = \mathbb{Y} \times \mathbb{Z} \times \mathbb{X}$,

$$L_\sigma(y, z; x) := f_1(y) + f_2(z) + \langle x, B_1 y + B_2 z - c \rangle + \frac{\sigma}{2} \|B_1 y + B_2 z - c\|^2.$$

The dual of problem (4.22) is given by

$$\max_{x \in \mathbb{X}} \{-f_1^*(-B_1^* x) - f_2^*(-B_2^* x) - \langle c, x \rangle\}, \quad (4.23)$$

where $B_1^* : \mathbb{X} \rightarrow \mathbb{Y}$ and $B_2^* : \mathbb{X} \rightarrow \mathbb{Z}$ are the adjoint of B_1 and B_2 , respectively. We first introduce some quantities for further analysis. Specifically, we consider the following self-adjoint linear operator $\mathcal{M} : \mathbb{W} \rightarrow \mathbb{W}$ defined by

$$\mathcal{M} = \begin{bmatrix} \sigma B_1^* B_1 + \mathcal{T}_1 & 0 & B_1^* \\ 0 & \mathcal{T}_2 & 0 \\ B_1 & 0 & \sigma^{-1} \mathcal{I} \end{bmatrix}. \quad (4.24)$$

Furthermore, since f_1 and f_2 are proper closed convex functions, there exist two self-adjoint and positive semidefinite operators Σ_{f_1} and Σ_{f_2} such that for all $y, \hat{y} \in \text{dom}(f_1)$, $\phi \in \partial f_1(y)$, and $\hat{\phi} \in \partial f_1(\hat{y})$,

$$f_1(y) \geq f_1(\hat{y}) + \langle \hat{\phi}, y - \hat{y} \rangle + \frac{1}{2} \|y - \hat{y}\|_{\Sigma_{f_1}}^2 \quad \text{and} \quad \langle \phi - \hat{\phi}, y - \hat{y} \rangle \geq \|y - \hat{y}\|_{\Sigma_{f_1}}^2,$$

and for all $z, \hat{z} \in \text{dom}(f_2)$, $\varphi \in \partial f_2(z)$, and $\hat{\varphi} \in \partial f_2(\hat{z})$,

$$f_2(z) \geq f_2(\hat{z}) + \langle \hat{\varphi}, z - \hat{z} \rangle + \frac{1}{2} \|z - \hat{z}\|_{\Sigma_{f_2}}^2 \quad \text{and} \quad \langle \varphi - \hat{\varphi}, z - \hat{z} \rangle \geq \|z - \hat{z}\|_{\Sigma_{f_2}}^2.$$

On the other hand, it follows from (Rockafellar, 1970, Corollary 28.3.1) that $(y^*, z^*) \in \mathbb{Y} \times \mathbb{Z}$ is an optimal solution to problem (4.22) if there exists $x^* \in \mathbb{X}$ such that (y^*, z^*, x^*) satisfies the following KKT system:

$$-B_1^*x^* \in \partial f_1(y^*), \quad -B_2^*x^* \in \partial f_2(z^*), \quad B_1y^* + B_2z^* - c = 0, \quad (4.25)$$

where ∂f_1 and ∂f_2 are the subdifferential mappings of f_1 and f_2 , respectively. Then, solving problem (4.22) is equivalent to finding $w \in \mathbb{W}$ such that $0 \in \mathcal{T}w$, where the maximal monotone operator $\mathcal{T}$ is defined by

$$\mathcal{T}w = \begin{pmatrix} \partial f_1(y) + B_1^*x \\ \partial f_2(z) + B_2^*x \\ c - B_1y - B_2z \end{pmatrix} \quad \forall w = (y, z, x) \in \mathbb{W}. \quad (4.26)$$

In this section, we make the following assumptions:

Assumption 3. *The KKT system (4.25) has a nonempty solution set.*

Assumption 4. *Both $\Sigma_{f_1} + B_1^*B_1 + \mathcal{T}_1$ and $\Sigma_{f_2} + B_2^*B_2 + \mathcal{T}_2$ are positive definite.*

Under Assumption 4 originated from (Fazel et al., 2013), each step of the pADMM is well-defined due to the strong convexity of the objective functions in the sub-problems. Furthermore, the following lemma shows that $(\mathcal{M} + \mathcal{T})^{-1}$ is Lipschitz continuous under Assumption 4.

Lemma 4.1. *Suppose that Assumption 4 holds. Consider the operators $\mathcal{T}$ defined in (4.26) and $\mathcal{M}$ defined in (4.24). Then, $(\mathcal{M} + \mathcal{T})^{-1}$ is Lipschitz continuous.*

Proof. We begin by establishing that $(\mathcal{M} + \mathcal{T})^{-1}$ is single-valued by proof of contradiction. Assume, for the sake of contradiction, that $(\mathcal{M} + \mathcal{T})^{-1}$ is not single-valued. Then, there exist distinct $\bar{w}_1 = (\bar{y}_1, \bar{z}_1, \bar{x}_1) \in \mathbb{W}$ and $\bar{w}_2 = (\bar{y}_2, \bar{z}_2, \bar{x}_2) \in \mathbb{W}$ such that $\bar{w}_1 \in (\mathcal{M} + \mathcal{T})^{-1}v$ and $\bar{w}_2 \in (\mathcal{M} + \mathcal{T})^{-1}v$ for some $v = (v_y, v_z, v_x) \in \mathbb{W}$, which

implies, for $i = 1, 2$,

$$v_y \in \partial f_1(\bar{y}_i) + (\sigma B_1^* B_1 + \mathcal{T}_1)\bar{y}_i + 2B_1^* \bar{x}_i, \quad (4.27)$$

$$v_z \in \partial f_2(\bar{z}_i) + B_2^* \bar{x}_i + \mathcal{T}_2 \bar{z}_i, \quad (4.28)$$

$$v_x = c - B_2 \bar{z}_i + \sigma^{-1} \bar{x}_i. \quad (4.29)$$

Using the definitions of Σ_{f_1} and Σ_{f_2} , we derive the following inequalities from equations (4.27), (4.28), and (4.29):

$$\langle -(\sigma B_1^* B_1 + \mathcal{T}_1)(\bar{y}_1 - \bar{y}_2) - 2B_1^*(\bar{x}_1 - \bar{x}_2), \bar{y}_1 - \bar{y}_2 \rangle \geq \|\bar{y}_1 - \bar{y}_2\|_{\Sigma_{f_1}}^2, \quad (4.30)$$

$$\langle -\mathcal{T}_2(\bar{z}_1 - \bar{z}_2) - B_2^*(\bar{x}_1 - \bar{x}_2), \bar{z}_1 - \bar{z}_2 \rangle \geq \|\bar{z}_1 - \bar{z}_2\|_{\Sigma_{f_2}}^2, \quad (4.31)$$

$$\sigma B_2(\bar{z}_1 - \bar{z}_2) = \bar{x}_1 - \bar{x}_2. \quad (4.32)$$

Substituting (4.32) into (4.31), we have

$$0 \geq \|\bar{z}_1 - \bar{z}_2\|_{\Sigma_{f_2} + \mathcal{T}_2 + \sigma B_2^* B_2}^2.$$

As $\Sigma_{f_2} + \mathcal{T}_2 + \sigma B_2^* B_2$ is positive definite by Assumption 4, we conclude that

$$\bar{z}_1 - \bar{z}_2 = 0, \quad (4.33)$$

which together with (4.32) yields that

$$\bar{x}_1 - \bar{x}_2 = 0. \quad (4.34)$$

Similarly, substituting (4.34) into (4.30), we can deduce from the positive definiteness of $\Sigma_{f_1} + \mathcal{T}_1 + \sigma B_1^* B_1$ by Assumption 4 that

$$\bar{y}_1 - \bar{y}_2 = 0. \quad (4.35)$$

Therefore, equations (4.33), (4.34), and (4.35) show that $\bar{w}_1 = \bar{w}_2$, contradicting our assumption. Thus, we conclude that $(\mathcal{M} + \mathcal{T})^{-1}$ is indeed single-valued.

To show the Lipschitz continuity of $(\mathcal{M} + \mathcal{T})^{-1}$, consider $\bar{w}_i = (\bar{y}_i, \bar{z}_i, \bar{x}_i) \in \mathbb{W}$ such that $\bar{w}_i = (\mathcal{M} + \mathcal{T})^{-1}v_i$, where $v_i = (v_{iy}, v_{iz}, v_{ix})$ for $i = 1, 2$. Similar to (4.27), (4.28), and (4.29), we have for $i = 1, 2$,

$$v_{iy} - 2B_1^* \bar{x}_i \in (\partial f_1 + \sigma B_1^* B_1 + \mathcal{T}_1) \bar{y}_i, \quad (4.36)$$

$$v_{iz} - \sigma B_2^* (v_{ix} - c) \in (\partial f_2 + \sigma B_2^* B_2 + \mathcal{T}_2) \bar{z}_i, \quad (4.37)$$

$$v_{ix} = c - B_2 \bar{z}_i + \sigma^{-1} \bar{x}_i. \quad (4.38)$$

By (Rockafellar, 1970, Corollary 23.5.1), we can obtain for $i = 1, 2$,

$$\begin{aligned} \bar{y}_i &\in (\partial f_1 + \sigma B_1^* B_1 + \mathcal{T}_1)^{-1}(v_{iy} - 2B_1^* \bar{x}_i), \\ \bar{z}_i &\in (\partial f_2 + \sigma B_2^* B_2 + \mathcal{T}_2)^{-1}(v_{iz} - \sigma B_2^* (v_{ix} - c)). \end{aligned}$$

Since $\partial f_2 + \sigma B_2^* B_2 + \mathcal{T}_2$ is strongly monotone by Assumption 4, $(\partial f_2 + \sigma B_2^* B_2 + \mathcal{T}_2)^{-1}$ is Lipschitz continuous by (Rockafellar and Wets, 1998, Proposition 12.54). Therefore, there exists a constant L_2 such that

$$\begin{aligned} \|\bar{z}_1 - \bar{z}_2\| &\leq L_2 \|(v_{1z} - v_{2z}) - \sigma B_2^* (v_{1x} - v_{2x})\| \\ &\leq L_2 \|v_{1z} - v_{2z}\| + \sigma L_2 \|B_2^*\| \|v_{1x} - v_{2x}\|. \end{aligned} \quad (4.39)$$

Hence, by (4.38) and (4.39), we can obtain

$$\begin{aligned} \|\bar{x}_1 - \bar{x}_2\| &= \|\sigma(v_{1x} - v_{2x} + B_2(\bar{z}_1 - \bar{z}_2))\| \\ &\leq \sigma(1 + \sigma L_2 \|B_2\| \|B_2^*\|) \|v_{1x} - v_{2x}\| + \sigma L_2 \|B_2\| \|v_{1z} - v_{2z}\|. \end{aligned} \quad (4.40)$$

Similarly, since $\partial f_1 + \sigma B_1^* B_1 + \mathcal{T}_1$ is strongly monotone by Assumption 4, there exists a constant L_1 such that

$$\begin{aligned} \|\bar{y}_1 - \bar{y}_2\| &\leq L_1 \|(v_{1y} - v_{2y}) - 2B_1^* (\bar{x}_1 - \bar{x}_2)\| \\ &\leq L_1 (\|v_{1y} - v_{2y}\| + 2\|B_1^*\| \|\bar{x}_1 - \bar{x}_2\|) \\ &\leq L_1 (\|v_{1y} - v_{2y}\| + 2\|B_1^*\| \sigma(1 + \sigma L_2 \|B_2\| \|B_2^*\|) \|v_{1x} - v_{2x}\| \\ &\quad + 2\|B_1^*\| \sigma L_2 \|B_2\| \|v_{1z} - v_{2z}\|). \end{aligned} \quad (4.41)$$

Therefore, by (4.39), (4.40), and (4.41), there exists a constant L such that $\|\bar{w}_1 - \bar{w}_2\| \leq L \|v_1 - v_2\|$. $\square$

Inspired by the interpretation of pADMM as a (partial) PPA in (Chen et al., 2015) (for $\rho = 1$) and (Yang et al., 2025) (for $\rho \in \mathbb{R}$), below we rigorously establish an important one-to-one correspondence between pADMM and dPPA.

Proposition 4.2. *Suppose that Assumption 4 holds. Consider the operators $\mathcal{T}$ defined in (4.26) and $\mathcal{M}$ defined in (4.24). Then the sequence $\{w^k\}$ generated by the pADMM in Algorithm 1 coincides with the sequence $\{w^k\}$ generated by the dPPA in (4.2) with the same initial point $w^0 \in \mathbb{W}$. Additionally, $\mathcal{M}$ is an admissible preconditioner such that $(\mathcal{M} + \mathcal{T})^{-1}$ is Lipschitz continuous.*

Proof. Similar to the proof establishing the equivalence of the pADMM and the (partial) PPA, as outlined in Appendix B of (Yang et al., 2025), we can obtain that under Assumption 4, the sequence $\{w^k\}$ generated by Algorithm 1 coincides with the sequence $\{w^k\}$ generated by the following scheme:

$$\mathcal{M}w^k \in (\mathcal{M} + \mathcal{T})\bar{w}^k, \quad w^{k+1} = (1 - \rho_k)w^k + \rho_k\bar{w}^k.$$

Since $(\mathcal{M} + \mathcal{T})^{-1}$ is single-valued according to Lemma 4.1, we have

$$\bar{w}^k = (\mathcal{M} + \mathcal{T})^{-1}\mathcal{M}w^k, \quad w^{k+1} = (1 - \rho_k)w^k + \rho_k\bar{w}^k.$$

Note that for an arbitrary choice of $w^k \in \mathbb{W}$, each step in Algorithm 1 is well-defined under Assumption 4. Hence, based on the equivalence established, we conclude that $(\mathcal{M} + \mathcal{T})^{-1}\mathcal{M}$ has full domain. Combining this result with Lemma 4.1, we deduce that $\mathcal{M}$ is an admissible preconditioner such that $(\mathcal{M} + \mathcal{T})^{-1}$ is Lipschitz continuous, which completes the proof. $\square$

The convergence of Algorithm 1 with the uniform relaxation factor $\rho \in (0, 2)$ has already been established in (Xiao et al., 2018) under Assumptions 3 and 4 by borrowing the proofs from (Fazel et al., 2013). In contrast, by leveraging the equivalence between the dPPA and the pADMM, as detailed in Proposition 4.2, we can directly

deduce the convergence of Algorithm 1 with varying relaxation factors $\rho_k \in (0, 2)$ for $k \geq 0$ by applying the convergence results of the dPPA, as presented in Theorem 2.1.

Corollary 4.2. *Suppose that Assumptions 3 and 4 hold. If $0 < \inf_k \rho_k \leq \sup_k \rho_k < 2$, then the sequence $\{w^k\} = \{(y^k, z^k, x^k)\}$ generated by Algorithm 1 converges to the point $w^* = (y^*, z^*, x^*)$, where (y^*, z^*) is a solution to problem (4.22) and x^* is a solution to problem (4.23).*

4.2.2 An Accelerated pADMM

Based on the equivalence stated in Proposition 4.2, we can employ the accelerated dPPA introduced in Algorithm 7 to derive an accelerated pADMM, as outlined in Algorithm 8.

Algorithm 8 An accelerated pADMM for solving the COP (4.22)

- 1: **Input:** Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two self-adjoint positive semidefinite linear operators on $\mathbb{Y}$ and $\mathbb{Z}$, respectively. Choose an initial point $w^0 = (y^0, z^0, x^0) \in \text{dom}(f_1) \times \text{dom}(f_2) \times \mathbb{X}$. Let $\hat{w}^0 := w^0$. Set parameters $\sigma > 0$, $\alpha \geq 2$, and $\rho \in (0, 2]$.
 - 2: **for** $k = 0, 1, \dots$ **do**
 - 3: Step 1. $\bar{z}^k = \arg \min_{z \in \mathbb{Z}} \{L_\sigma(y^k, z; x^k) + \frac{1}{2}\|z - z^k\|_{\mathcal{T}_2}^2\}$
 - 4: Step 2. $\bar{x}^k = x^k + \sigma(B_1 y^k + B_2 \bar{z}^k - c)$
 - 5: Step 3. $\bar{y}^k = \arg \min_{y \in \mathbb{Y}} \{L_\sigma(y, \bar{z}^k; \bar{x}^k) + \frac{1}{2}\|y - y^k\|_{\mathcal{T}_1}^2\}$
 - 6: Step 4. $\hat{w}^{k+1} = (1 - \rho)w^k + \rho \bar{w}^k$
 - 7: Step 5. $w^{k+1} = w^k + \frac{\alpha}{2(k+\alpha)}(\hat{w}^{k+1} - w^k) + \frac{k}{k+\alpha}(\hat{w}^{k+1} - \hat{w}^k)$
-

Remark 4.4. *When $\mathcal{T}_i = 0$ for $i = 1, 2$ in Algorithm 8, we can obtain an accelerated ADMM. As will be shown in Proposition 4.3, this accelerated ADMM with $\alpha = 2$ in Algorithm 8 is equivalent to the HPR in Algorithm 4. Furthermore, by setting $\mathcal{T}_i = \sigma(\lambda_{\max}(B_i^* B_i)\mathcal{I} - B_i^* B_i)$ for $i = 1, 2$ in Algorithm 8, we can obtain an accelerated LADMM. Compared to the algorithm in (Li and Lin, 2019), the $\mathcal{T}_i$ for $i = 1, 2$ in Algorithm 8 will not tend to infinity as k increases, which implies that this*

accelerated LADMM has a larger primal step length. Additionally, both $\mathcal{T}_1$ and $\mathcal{T}_2$ in Algorithm 8 can be positive semidefinite under Assumption 4, which is a significant difference compared to the work of (Sabach and Teboulle, 2022). Finally, the accelerated pADMM introduced in (Yang et al., 2025), where both $\mathcal{T}_1$ and $\mathcal{T}_2$ are positive definite, is a special case of Algorithm 8 with $\alpha = 2$.

According to the convergence result of the accelerated dPPA in Theorem 4.1, we can obtain the global convergence of Algorithm 8 in the following corollary.

Corollary 4.3. *Suppose that Assumptions 3 and 4 hold. The sequence $\{\bar{w}^k\} = \{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ generated by Algorithm 8 converges to the point $w^* = (y^*, z^*, x^*)$, where (y^*, z^*) is a solution to problem (4.22) and x^* is a solution to problem (4.23).*

Proof. The proof follows from Theorem 4.1 and Proposition 4.2. $\square$

To analyze the convergence rate of Algorithm 8, we begin by considering the residual mapping associated with the KKT system (4.25), as introduced in (Han et al., 2018):

$$\mathcal{R}(w) := \begin{pmatrix} y - \text{Prox}_{f_1}(y - B_1^*x) \\ z - \text{Prox}_{f_2}(z - B_2^*x) \\ c - B_1y - B_2z \end{pmatrix} \quad \forall w = (y, z, x) \in \mathbb{W}. \quad (4.42)$$

It is clear that $w^* = (y^*, z^*, x^*)$ satisfies the KKT system (4.25) if and only if $\mathcal{R}(w^*) = 0$. Let $\{(\bar{y}^k, \bar{z}^k)\}$ be the sequence generated by Algorithm 8. To estimate the objective error, we define

$$h(\bar{y}^k, \bar{z}^k) := f_1(\bar{y}^k) + f_2(\bar{z}^k) - f_1(y^*) - f_2(z^*) \quad \forall k \geq 0,$$

where (y^*, z^*) is the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k)\}$. The next lemma provides the lower and upper bounds for the objective error.

Lemma 4.2. *Suppose that Assumptions 3 and 4 hold. Let $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ be the sequence generated by Algorithm 8. Then for all $k \geq 0$, we have the following bounds:*

$$h(\bar{y}^k, \bar{z}^k) \leq \langle \sigma B_1(y^* - \bar{y}^k) - \bar{x}^k, (B_1 \bar{y}^k + B_2 \bar{z}^k - c) \rangle + \langle y^* - \bar{y}^k, \mathcal{T}_1(\bar{y}^k - y^k) \rangle + \langle z^* - \bar{z}^k, \mathcal{T}_2(\bar{z}^k - z^k) \rangle \quad (4.43)$$

and

$$h(\bar{y}^k, \bar{z}^k) \geq \langle B_1 \bar{y}^k + B_2 \bar{z}^k - c, -x^* \rangle, \quad (4.44)$$

where (y^*, z^*, x^*) is the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$.

Proof. For notational convenience, for any $x \in \mathbb{X}$, define

$$d_{f_1}(x) := f_1^*(-B_1^*x) + \langle x, c \rangle, \quad d_{f_2}(x) := f_2^*(-B_2^*x)$$

and

$$\tilde{x}^k := \bar{x}^k + \sigma(B_1 \bar{y}^k + B_2 \bar{z}^k - c) \quad \forall k \geq 0.$$

On the one hand, from the optimality conditions of the subproblems of Algorithm 8, we have for any $k \geq 0$,

$$-B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k) \in \partial f_2(\bar{z}^k), \quad -B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k) \in \partial f_1(\bar{y}^k).$$

It follows from (Rockafellar, 1970, Theorem 23.5) that for any $k \geq 0$,

$$\begin{cases} f_1^*(-B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k)) = \langle \bar{y}^k, -B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k) \rangle - f_1(\bar{y}^k), \\ f_2^*(-B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)) = \langle \bar{z}^k, -B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k) \rangle - f_2(\bar{z}^k). \end{cases}$$

Summing them up, we can obtain that for any $k \geq 0$,

$$\begin{aligned} & f_1^*(-B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k)) + f_2^*(-B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)) + \langle \tilde{x}^k, c \rangle \\ = & -f_1(\bar{y}^k) - f_2(\bar{z}^k) - \langle (B_1 \bar{y}^k + B_2 \bar{z}^k - c), \bar{x}^k + \sigma(B_1 \bar{y}^k - c) \rangle \\ & - \langle \bar{y}^k, \mathcal{T}_1(\bar{y}^k - y^k) \rangle - \langle \bar{z}^k, \mathcal{T}_2(\bar{z}^k - z^k) \rangle. \end{aligned} \quad (4.45)$$

On the other hand, from the KKT conditions in (4.25) and (Rockafellar, 1970, Theorem 23.5), we have

$$y^* \in \partial f_1^*(-B_1^* x^*), \quad z^* \in \partial f_2^*(-B_2^* x^*).$$

Thus, it follows from the convexity of f_1^* and f_2^* that for any $k \geq 0$,

$$\left\{ \begin{array}{l} f_1^* (-B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k)) + \langle \tilde{x}^k, c \rangle - d_{f_1}(x^*) \\ \geq \langle y^*, -B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k) + B_1^* x^* \rangle + \langle c, \tilde{x}^k - x^* \rangle, \\ f_2^* (-B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)) - d_{f_2}(x^*) \geq \langle z^*, -B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k) + B_2^* x^* \rangle. \end{array} \right.$$

Summing them up and by noting $B_1 y^* + B_2 z^* = c$, we have for any $k \geq 0$,

$$\begin{aligned} & f_1^* (-B_1^* \tilde{x}^k - \mathcal{T}_1(\bar{y}^k - y^k)) + f_2^* (-B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)) + \langle \tilde{x}^k, c \rangle \\ & \geq d_{f_1}(x^*) + d_{f_2}(x^*) - \langle B_1 y^* - c, \sigma(B_1 \bar{y}^k + B_2 \bar{z}^k - c) \rangle \\ & \quad - \langle y^*, \mathcal{T}_1(\bar{y}^k - y^k) \rangle - \langle z^*, \mathcal{T}_2(\bar{z}^k - z^k) \rangle. \end{aligned} \quad (4.46)$$

Combining (4.45) and (4.46), we can deduce that for any $k \geq 0$,

$$\begin{aligned} & f_1(\bar{y}^k) + f_2(\bar{z}^k) + d_{f_1}(x^*) + d_{f_2}(x^*) \\ & \leq \langle \sigma B_1(y^* - \bar{y}^k) - \bar{x}^k, (B_1 \bar{y}^k + B_2 \bar{z}^k - c) \rangle \\ & \quad + \langle y^* - \bar{y}^k, \mathcal{T}_1(\bar{y}^k - y^k) \rangle + \langle z^* - \bar{z}^k, \mathcal{T}_2(\bar{z}^k - z^k) \rangle. \end{aligned}$$

Since $d_{f_1}(x^*) + d_{f_2}(x^*) = -f_1(y^*) - f_2(z^*)$ from (Rockafellar, 1970, Theorem 28.4),

we can derive for any $k \geq 0$,

$$\begin{aligned} h(\bar{y}^k, \bar{z}^k) & \leq \langle \sigma B_1(y^* - \bar{y}^k) - \bar{x}^k, (B_1 \bar{y}^k + B_2 \bar{z}^k - c) \rangle \\ & \quad + \langle y^* - \bar{y}^k, \mathcal{T}_1(\bar{y}^k - y^k) \rangle + \langle z^* - \bar{z}^k, \mathcal{T}_2(\bar{z}^k - z^k) \rangle. \end{aligned}$$

Finally, according to the KKT conditions in (4.25), we have for any $k \geq 0$,

$$f_1(\bar{y}^k) - f_1(y^*) \geq \langle -B_1^* x^*, \bar{y}^k - y^* \rangle, \quad f_2(\bar{z}^k) - f_2(z^*) \geq \langle -B_2^* x^*, \bar{z}^k - z^* \rangle.$$

Thus, the inequality (4.44) holds since $B_1 y^* + B_2 z^* = c$. This completes the proof of the lemma. $\square$

Now, we are ready to present the complexity result for Algorithm 8.

Theorem 4.2. *Suppose that Assumptions 3 and 4 hold. Let $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ be the sequence generated by Algorithm 8, and let $w^* = (y^*, z^*, x^*)$ be the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ and $R_0 = \|w^0 - w^*\|_{\mathcal{M}}$.*

(a) If $\alpha = 2$, then for all $k \geq 0$, we have the following bounds:

$$\|\mathcal{R}(\bar{w}^k)\| \leq \left(\frac{\sigma \|B_1^*\| + 1}{\sqrt{\sigma}} + \|\sqrt{\mathcal{T}_2}\| + \|\sqrt{\mathcal{T}_1}\| \right) \frac{2R_0}{\rho(k+1)} \quad (4.47)$$

and

$$\left(\frac{-1}{\sqrt{\sigma}} \|x^*\| \right) \frac{2R_0}{\rho(k+1)} \leq h(\bar{y}^k, \bar{z}^k) \leq \left(3R_0 + \frac{1}{\sqrt{\sigma}} \|x^*\| \right) \frac{2R_0}{\rho(k+1)}. \quad (4.48)$$

(b) If $\alpha > 2$, then we have the following bounds:

$$\|\mathcal{R}(\bar{w}^k)\| = \left(\frac{\sigma \|B_1^*\| + 1}{\sqrt{\sigma}} + \|\sqrt{\mathcal{T}_2}\| + \|\sqrt{\mathcal{T}_1}\| \right) o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty \quad (4.49)$$

and

$$|h(\bar{y}^k, \bar{z}^k)| = o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty. \quad (4.50)$$

Proof. Consider the case where $\alpha = 2$. We first estimate the convergence rate of $\mathcal{R}(\bar{w}^k)$ for any $k \geq 0$. According to Proposition 4.1, we have

$$\|\hat{w}^{k+1} - w^k\|_{\mathcal{M}}^2 \leq \frac{4R_0^2}{(k+1)^2} \quad \forall k \geq 0.$$

By the definition of $\mathcal{M}$ in (4.24), this can be rewritten as

$$\|\hat{y}^{k+1} - y^k\|_{\mathcal{T}_1}^2 + \frac{1}{\sigma} \|\sigma B_1(\hat{y}^{k+1} - y^k) + (\hat{x}^{k+1} - x^k)\|^2 + \|\hat{z}^{k+1} - z^k\|_{\mathcal{T}_2}^2 \leq \frac{4R_0^2}{(k+1)^2} \quad \forall k \geq 0. \quad (4.51)$$

Also, from Step 4 in Algorithm 8, we can obtain that for any $k \geq 0$,

$$\begin{cases} \hat{y}^{k+1} - y^k = \rho(\bar{y}^k - y^k), \\ \hat{z}^{k+1} - z^k = \rho(\bar{z}^k - z^k), \\ \hat{x}^{k+1} - x^k = \rho(\bar{x}^k - x^k). \end{cases}$$

Thus, we can rewrite (4.51) as

$$\|\bar{y}^k - y^k\|_{\mathcal{T}_1}^2 + \frac{1}{\sigma} \|\sigma B_1(\bar{y}^k - y^k) + (\bar{x}^k - x^k)\|^2 + \|\bar{z}^k - z^k\|_{\mathcal{T}_2}^2 \leq \frac{4R_0^2}{\rho^2(k+1)^2} \quad \forall k \geq 0. \quad (4.52)$$

Due to Step 2 in Algorithm 8, we can deduce that for any $k \geq 0$,

$$\begin{aligned}\|\sigma B_1(\bar{y}^k - y^k) + (\bar{x}^k - x^k)\| &= \|\sigma B_1(\bar{y}^k - y^k) + \sigma(B_1 y^k + B_2 \bar{z}^k - c)\| \\ &= \sigma \|B_1 \bar{y}^k + B_2 \bar{z}^k - c\|,\end{aligned}$$

which together with (4.52) yields that

$$\|B_1 \bar{y}^k + B_2 \bar{z}^k - c\| \leq \frac{2R_0}{\sqrt{\sigma}\rho(k+1)} \quad \forall k \geq 0. \quad (4.53)$$

Moreover, from the optimality conditions of the subproblems in Algorithm 8, we have for any $k \geq 0$,

$$\begin{cases} \bar{z}^k = \text{Prox}_{f_2}(\bar{z}^k - B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)), \\ \bar{y}^k = \text{Prox}_{f_1}(\bar{y}^k - B_1^*(\bar{x}^k + \sigma(B_1 \bar{y}^k + B_2 \bar{z}^k - c)) - \mathcal{T}_1(\bar{y}^k - y^k)), \end{cases} \quad (4.54)$$

which together with (4.52) yields that for any $k \geq 0$,

$$\begin{aligned}& \|\bar{z}^k - \text{Prox}_{f_2}(\bar{z}^k - B_2^* \bar{x}^k)\| \\ &= \|\text{Prox}_{f_2}(\bar{z}^k - B_2^* \bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)) - \text{Prox}_{f_2}(\bar{z}^k - B_2^* \bar{x}^k)\| \\ &\leq \|\mathcal{T}_2(\bar{z}^k - z^k)\| \\ &\leq \|\sqrt{\mathcal{T}_2}\| \|\bar{z}^k - z^k\|_{\mathcal{T}_2} \\ &\leq \|\sqrt{\mathcal{T}_2}\| \frac{2R_0}{\rho(k+1)}.\end{aligned} \quad (4.55)$$

Similarly, from (4.52), (4.53), and (4.54), we also have for any $k \geq 0$,

$$\begin{aligned}& \|\bar{y}^k - \text{Prox}_{f_1}(\bar{y}^k - B_1^* \bar{x}^k)\| \\ &\leq \|B_1^* \sigma(B_1 \bar{y}^k + B_2 \bar{z}^k - c) + \mathcal{T}_1(\bar{y}^k - y^k)\| \\ &\leq \sigma \|B_1^*\| \|B_1 \bar{y}^k + B_2 \bar{z}^k - c\| + \|\mathcal{T}_1(\bar{y}^k - y^k)\| \\ &\leq (\sqrt{\sigma} \|B_1^*\| + \|\sqrt{\mathcal{T}_1}\|) \frac{2R_0}{\rho(k+1)}.\end{aligned} \quad (4.56)$$

Therefore, by (4.53), (4.55), and (4.56), we can obtain that for any $k \geq 0$,

$$\begin{aligned}\|\mathcal{R}(\bar{w}^k)\| &\leq \sqrt{\left(\frac{1}{\sigma} + \|\sqrt{\mathcal{T}_2}\|^2 + (\sqrt{\sigma} \|B_1^*\| + \|\sqrt{\mathcal{T}_1}\|)^2\right) \frac{2R_0}{\rho(k+1)}} \\ &\leq \left(\frac{\sigma \|B_1^*\| + 1}{\sqrt{\sigma}} + \|\sqrt{\mathcal{T}_2}\| + \|\sqrt{\mathcal{T}_1}\|\right) \frac{2R_0}{\rho(k+1)}.\end{aligned} \quad (4.57)$$

Now, we estimate the complexity result concerning the primal function value gap. For the lower bound of the primal function value gap, from (4.44) and (4.53), we have for all $k \geq 0$,

$$\begin{aligned} h(\bar{y}^k, \bar{z}^k) &\geq \langle B_1 \bar{y}^k + B_2 \bar{z}^k - c, -x^* \rangle \\ &\geq -\|x^*\| \|B_1 \bar{y}^k + B_2 \bar{z}^k - c\| \\ &\geq -\frac{2R_0 \|x^*\|}{\sqrt{\sigma} \rho(k+1)}. \end{aligned}$$

On the other hand, $\mathcal{F}_{\mathcal{T}, \rho}^{\mathcal{M}}$ is $\mathcal{M}$ -nonexpansive for $\rho \in (0, 2]$ by Proposition 2.2. Similar to (4.20), we can derive from the induction that

$$\|w^k - w^*\|_{\mathcal{M}} \leq R_0 \quad \forall k \geq 0.$$

Hence, from the $\mathcal{M}$ -nonexpansiveness of $\mathcal{F}_{\mathcal{T}}^{\mathcal{M}}$ by Proposition 2.2, we have

$$\|\bar{w}^k - w^*\|_{\mathcal{M}} = \|\mathcal{F}_{\mathcal{T}}^{\mathcal{M}} w^k - w^*\|_{\mathcal{M}} \leq \|w^k - w^*\|_{\mathcal{M}} \leq R_0 \quad \forall k \geq 0,$$

which implies

$$\|\bar{y}^k - y^*\|_{\mathcal{T}_1}^2 + \frac{1}{\sigma} \|\sigma B_1(\bar{y}^k - y^*) + (\bar{x}^k - x^*)\|^2 + \|\bar{z}^k - z^*\|_{\mathcal{T}_2}^2 \leq R_0^2 \quad \forall k \geq 0.$$

This inequality together with (4.43), (4.52), and (4.53) yields that for all $k \geq 0$,

$$\begin{aligned} h(\bar{y}^k, \bar{z}^k) &\leq (\|\sigma B_1(\bar{y}^k - y^*) + (\bar{x}^k - x^*)\| + \|x^*\|) \|B_1 \bar{y}^k + B_2 \bar{z}^k - c\| \\ &\quad + \|y^* - \bar{y}^k\|_{\mathcal{T}_1} \|\bar{y}^k - y^k\|_{\mathcal{T}_1} + \|z^* - \bar{z}^k\|_{\mathcal{T}_2} \|\bar{z}^k - z^k\|_{\mathcal{T}_2} \\ &\leq (\sqrt{\sigma} R_0 + \|x^*\|) \frac{2R_0}{\sqrt{\sigma} \rho(k+1)} + \frac{4R_0^2}{\rho(k+1)} \\ &= \left(3R_0 + \frac{1}{\sqrt{\sigma}} \|x^*\|\right) \frac{2R_0}{\rho(k+1)}. \end{aligned}$$

Now, we establish the complexity results for the case where $\alpha > 2$. According to Proposition 4.1, we have

$$\|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} = o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty.$$

Similar to the previous case with $\alpha = 2$, we can obtain

$$\|\mathcal{R}(\bar{w}^k)\| = \left(\frac{\sigma\|B_1^*\|+1}{\sqrt{\sigma}} + \|\sqrt{\mathcal{T}_2}\| + \|\sqrt{\mathcal{T}_1}\| \right) o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty.$$

In addition, since $\|\bar{w}^k - w^*\|$ is bounded due to Corollary 4.3, from (4.43), there exists a constant C_1 such that

$$h(\bar{y}^k, \bar{z}^k) \leq C_1 \left(\|B_1\bar{y}^k + B_2\bar{z}^k - c\| + \|\bar{y}^k - y^k\|_{\mathcal{T}_1} + \|\bar{z}^k - z^k\|_{\mathcal{T}_2} \right).$$

Hence, following the proof of the previous case with $\alpha = 2$, we can similarly demonstrate that

$$|h(\bar{y}^k, \bar{z}^k)| = o\left(\frac{1}{k+1}\right) \text{ as } k \rightarrow +\infty,$$

which completes the proof. $\square$

Remark 4.5. For convex optimization problem (4.22), $f_1(\cdot)$ and $f_2(\cdot)$ are often of a composite structure, i.e. $f_1 = g_1 + p_1$, and $f_2 = g_2 + p_2$, where $g_1 : \mathbb{Y} \rightarrow (-\infty, +\infty)$ and $g_2 : \mathbb{Z} \rightarrow (-\infty, +\infty)$ are two continuously differentiable convex functions (e.g., convex quadratic functions), and $p_1 : \mathbb{Y} \rightarrow (-\infty, +\infty]$ and $p_2 : \mathbb{Z} \rightarrow (-\infty, +\infty]$ are two proper closed convex functions. In this case, it is more convenient for computational purposes to define the following KKT residual mapping:

$$\tilde{\mathcal{R}}(w) := \begin{pmatrix} y - \text{Prox}_{p_1}(y - \nabla g_1(y) - B_1^*x) \\ z - \text{Prox}_{p_2}(z - \nabla g_2(z) - B_2^*x) \\ c - B_1y - B_2z \end{pmatrix} \quad \forall w = (y, z, x) \in \mathbb{W}. \quad (4.58)$$

By substituting $\tilde{\mathcal{R}}(\cdot)$ into (4.55) and (4.56), and replacing (4.54) with the relations:

$$\begin{cases} \bar{z}^k = \text{Prox}_{p_2}(\bar{z}^k - \nabla g_2(\bar{z}^k) - B_2^*\bar{x}^k - \mathcal{T}_2(\bar{z}^k - z^k)), \\ \bar{y}^k = \text{Prox}_{p_1}(\bar{y}^k - \nabla g_1(\bar{y}^k) - B_1^*(\bar{x}^k + \sigma(B_1\bar{y}^k + B_2\bar{z}^k - c)) - \mathcal{T}_1(\bar{y}^k - y^k)), \end{cases}$$

one can verify in a straightforward way that the iteration complexity results in Theorem 4.2 remain valid for $\tilde{\mathcal{R}}(\cdot)$.

Now, we aim to establish the equivalence between the HPR method without proximal terms in Algorithm 4 and the Halpern accelerating pADMM ($\alpha = 2$) without proximal terms in Algorithm 8 for solving the COP (4.22).

Proposition 4.3. *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 $w^0 = (y^0, z^0, x^0) \in \text{dom}(f_1) \times \text{dom}(f_2) \times \mathbb{X}$. Then, the sequence $\{(y^{k+1}, z^{k+1}, x^{k+1/2})\}$ generated by the HPR method without proximal terms in Algorithm 4, starting from the same initial point $(y^0, x^0) \in \text{dom}(f_1) \times \mathbb{X}$, is equivalent to the sequence $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ produced by the Halpern accelerating pADMM ($\alpha = 2, \rho = 2$) without proximal terms in Algorithm 8.*

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 (Bauschke and Combettes, 2017, 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:

$$z^1 = \bar{z}^0, \quad x^{1/2} = \bar{x}^0, \quad \text{and} \quad y^1 = \bar{y}^0. \quad (4.59)$$

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

$$\begin{aligned} \tilde{x}^1 &= \frac{1}{2}(\tilde{x}^0 + x^1) + \frac{\sigma}{2} B_1(y^0 - y^1) \\ &= \frac{1}{2}(\tilde{x}^0 + x^{1/2} + \sigma(B_1 y^1 + B_2 z^1 - c)) + \frac{\sigma}{2} B_1(y^0 - y^1) \\ &= \frac{1}{2}(x^{1/2} + \tilde{x}^0 + \sigma(B_1 y^0 + B_2 z^1 - c)) \\ &= x^{1/2}. \end{aligned} \quad (4.60)$$

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

$$\begin{aligned} 0 &\in \partial f_2(z^2) + B_2^* \tilde{x}^1 + \sigma B_2^*(B_1 y^1 + B_2 z^2 - c) \\ \iff 0 &\in \partial f_2(z^2) + B_2^*(x^{1/2} + \sigma(B_1 y^1 + B_2 z^2 - c)). \end{aligned} \quad (4.61)$$

On the other hand, $\bar{z}^1$ obtained by Step 1 in Algorithm 8 should satisfy the following optimality condition:

$$0 \in \partial f_2(\bar{z}^1) + B_2^*(x^1 + \sigma(B_1 y^1 + B_2 \bar{z}^1 - c)). \quad (4.62)$$

Substituting

$$x^1 = \frac{1}{2}x^0 + \frac{1}{2}(2\bar{x}^0 - x^0) \quad \text{and} \quad y^1 = \frac{1}{2}y^0 + \frac{1}{2}(2\bar{y}^0 - y^0) \quad (4.63)$$

from Steps 4 and 5 in Algorithm 8 into (4.62), we obtain:

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

This, together with (4.59) and (4.61), implies:

$$z^2 = \bar{z}^1. \quad (4.64)$$

Furthermore, according to Step 2 in Algorithm 4 and (4.60), we know that:

$$x^{3/2} = \tilde{x}^1 + \sigma(B_1y^1 + B_2z^2 - c) = x^{1/2} + \sigma(B_1y^1 + B_2z^2 - c). \quad (4.65)$$

Also, from Step 2 in Algorithm 8 and (4.63), we know that $\bar{x}^1$ satisfies:

$$\begin{aligned} \bar{x}^1 &= x^1 + \sigma(B_1y^1 + B_2\bar{z}^1 - c) \\ &= \bar{x}^0 + \sigma(B_1\bar{y}^0 + B_2\bar{z}^1 - c). \end{aligned}$$

It follows from (4.59), (4.64), and (4.65) that:

$$x^{3/2} = \bar{x}^1, \quad (4.66)$$

which, together with (4.64), implies:

$$y^2 = \bar{y}^1 \quad (4.67)$$

by comparing Step 3 in Algorithms 4 and 8. 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

$$y^{k+1} = \bar{y}^k, \quad z^{k+1} = \bar{z}^k, \quad \text{and} \quad x^{k+1/2} = \bar{x}^k. \quad (4.68)$$

According to Step 5 in Algorithm 4 and (4.68), we know that $\tilde{x}^{k+1}$ satisfies

$$\begin{aligned}\tilde{x}^{k+1} &= \left(\frac{1}{k+2}\tilde{x}^0 + \frac{k+1}{k+2}x^{k+1}\right) + \frac{\sigma}{k+2}[B_1y^0 - B_1y^{k+1}] \\ &= \frac{1}{k+2}\tilde{x}^0 + \frac{k+1}{k+2}(x^{k+1/2} + \sigma(B_1y^{k+1} + B_2z^{k+1} - c)) + \frac{\sigma}{k+2}B_1(y^0 - y^{k+1}) \\ &= \frac{1}{k+2}x^0 + \frac{k+1}{k+2}(\bar{x}^k + \sigma(B_1\bar{y}^k + B_2\bar{z}^k - c)) + \frac{\sigma}{k+2}B_1(y^0 - \bar{y}^k).\end{aligned}\tag{4.69}$$

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

$$0 \in \partial f_2(z^{k+2}) + B_2^*\tilde{x}^{k+1} + \sigma B_2^*(B_1y^{k+1} + B_2z^{k+2} - c).\tag{4.70}$$

On the other hand, from Steps 2 and 5 in Algorithm 8, we know that x^{k+1} satisfies

$$\begin{aligned}x^{k+1} &= \frac{1}{k+2}x^0 + \frac{k+1}{k+2}(2\bar{x}^k - x^k) \\ &= \frac{1}{k+2}x^0 + \frac{k+1}{k+2}(\bar{x}^k + \sigma(B_1y^k + B_2\bar{z}^k - c))\end{aligned}\tag{4.71}$$

and y^{k+1} satisfies

$$y^{k+1} = \frac{1}{k+2}y^0 + \frac{k+1}{k+2}(2\bar{y}^k - y^k).\tag{4.72}$$

In addition, from Step 1 in Algorithm 8, we have

$$0 \in \partial f_2(\bar{z}^{k+1}) + B_2^*(x^{k+1} + \sigma(B_1y^{k+1} + B_2\bar{z}^{k+1} - c)).\tag{4.73}$$

Substituting (4.71) and (4.72) into (4.73), we can obtain

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

which together with (4.69) implies

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

It follows from (4.68) and (4.70) that

$$z^{k+2} = \bar{z}^{k+1}.\tag{4.74}$$

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

$$x^{k+3/2} = \tilde{x}^{k+1} + \sigma(B_1y^{k+1} + B_2z^{k+2} - c).\tag{4.75}$$

Also, from Step 2 in Algorithm 8, (4.69), (4.71) and (4.72), we know that $\bar{x}^{k+1}$ satisfies

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

It follows from (4.68), (4.74), and (4.75) that

$$x^{k+3/2} = \bar{x}^{k+1}. \quad (4.76)$$

Since $(\bar{z}^{k+1}, \bar{x}^{k+1}) = (z^{k+2}, x^{k+3/2})$, then we have

$$y^{k+2} = \bar{y}^{k+1} \quad (4.77)$$

by comparing Step 3 in Algorithms 4 and 8. Hence, from (4.74), (4.76), and (4.77), we know that the statement also holds for $k + 1$. Thus, by induction, we have completed the proof. $\square$

4.3 A Halpern Accelerating Algorithm for Solving Reduced OT Problems

This section proposes HOT, an efficient Halpern-accelerated algorithm for solving discrete OT problems with finite supports in $\mathbb{R}^2$ under the L_2^2 -ground cost.

4.3.1 An Equivalent Reduced Model of the OT Problem

In the following discussion, we assume two-dimensional histograms for simplicity. As previously mentioned, these histograms are widely used in applications as shape and image descriptors. Without loss of generality, we adopt the following assumptions and notations:

1. Histograms have supports in $m = d_1 \times d_2$ bins with d_1 rows and d_2 columns;
2. The index set for bins is defined as $\mathcal{I} = \{(i, j) \mid 1 \leq i \leq d_1, 1 \leq j \leq d_2\}$. We use (i, j) to denote a bin or a node corresponding to it;

3. μ^1 and μ^2 are the two histograms to be compared, where each histogram μ^k is defined as $\{\mu_{i,j}^k \mid \mu_{i,j}^k \geq 0, (i,j) \in \mathcal{I}, \sum_{(i,j) \in \mathcal{I}} \mu_{i,j}^k = 1\}$ for $k = 1, 2$.

With these notations and assumptions, the discrete OT problem for computing the KW distance between histograms μ^1 and μ^2 can be defined as follows:

$$\begin{aligned} \min_{\pi} \quad & \sum_{(i,j) \in \mathcal{I}} \sum_{(k,l) \in \mathcal{I}} c_{i,j;k,l} \pi_{i,j;k,l} \\ \text{s.t.} \quad & \begin{cases} \sum_{(k,l) \in \mathcal{I}} \pi_{i,j;k,l} = \mu_{i,j}^1 & \forall (i,j) \in \mathcal{I}, \\ \sum_{(i,j) \in \mathcal{I}} \pi_{i,j;k,l} = \mu_{k,l}^2 & \forall (k,l) \in \mathcal{I}, \\ \pi_{i,j;k,l} \geq 0 & \forall (i,j) \in \mathcal{I} \forall (k,l) \in \mathcal{I}, \end{cases} \end{aligned} \quad (4.78)$$

where π is the transport plan between histograms μ^1 and μ^2 . The ground distance $c_{i,j;k,l}$ is commonly defined by the L_p^p distance:

$$c_{i,j;k,l} = \|(i,j)^\top - (k,l)^\top\|_p^p = (|i-k|^p + |j-l|^p). \quad (4.79)$$

In this thesis, we focus on the case where $p = 2$. By exploiting the separable structure of the ground distance, Auricchio et al. (2018) proposed the following equivalent model in terms of the optimal objective function value:

$$\begin{aligned} \min_{f^{(1)}, f^{(2)}} \quad & \sum_{(i,j) \in \mathcal{I}} \left[\sum_{k=1}^{d_1} (k-i)^2 f_{i,k,j}^{(1)} + \sum_{l=1}^{d_2} (j-l)^2 f_{k,j,l}^{(2)} \right] \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^{d_1} f_{i,k,j}^{(1)} = \sum_{l=1}^{d_2} f_{k,j,l}^{(2)} & \forall (k,j) \in \mathcal{I}, \\ \sum_{k=1}^{d_1} f_{i,k,j}^{(1)} = \mu_{i,j}^1 & \forall (i,j) \in \mathcal{I}, \\ \sum_{j=1}^{d_2} f_{k,j,l}^{(2)} = \mu_{k,l}^2 & \forall (k,l) \in \mathcal{I}, \\ f_{i,k,j}^{(1)} \geq 0, f_{k,j,l}^{(2)} \geq 0 & \forall (i,j), (k,l) \in \mathcal{I}, \end{cases} \end{aligned} \quad (4.80)$$

where $f_{i,k,j}^{(1)}$ denotes the input flow from bin (i,j) to (k,j) and $f_{k,j,l}^{(2)}$ denotes the output flow from bin (k,j) to (k,l) . Compared to formulation (4.78), formulation

(4.80) offers substantial computational benefits. Specifically, the reduced problem (4.80) only has $d_1 d_2^2 + d_1^2 d_2$ variables, whereas the original model has $d_1^2 d_2^2$ variables. Moreover, the reduced model remains an LP problem. Consequently, popular algorithms for LP problems can be applied to solve this reduced model, such as the network-simplex method and the interior point method. Although the computation and memory costs of these mentioned algorithms are lower for the reduced model, it remains a challenge for solving large-scale problems. In this thesis, we focus on addressing the challenges by designing a fast algorithm to solve the reduced model (4.80).

To facilitate the design of the algorithm, we reformulate the model (4.80) into the following standard form of LP:

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & \langle c, x \rangle + \delta_{\mathbb{R}_+^N}(x) \\ \text{s.t.} \quad & \hat{A}x = \hat{b}, \end{aligned} \tag{4.81}$$

where

1. $M = 3m - 1, N = d_1^2 d_2 + d_1 d_2^2$;
2. $x = [f^{(1)}; f^{(2)}] \in \mathbb{R}^{d_1^2 d_2} \times \mathbb{R}^{d_1 d_2^2}$ with

$$\begin{cases} f^{(1)} = \{f_{i,k,j}^{(1)}, \forall (i,j) \in \mathcal{I}, k = 1, \dots, d_1\}, \\ f^{(2)} = \{f_{k,j,l}^{(2)}, \forall (k,l) \in \mathcal{I}, j = 1, \dots, d_2\}; \end{cases}$$

3. $c = [c^1; c^2] \in \mathbb{R}^{d_1^2 d_2} \times \mathbb{R}^{d_1 d_2^2}$ with

$$\begin{cases} c^1 = \{c_{i,k,j}^{(1)} = (k - i)^2, \forall (i,j) \in \mathcal{I}, k = 1, \dots, d_1\}, \\ c^2 = \{c_{k,j,l}^{(2)} = (j - l)^2, \forall (k,l) \in \mathcal{I}, j = 1, \dots, d_2\}; \end{cases}$$

4. $\hat{b} = [\mathbf{0}_m; \mu^1; \mu^2] \in \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m$;

$$5. \hat{A} = \begin{bmatrix} A_1 & A_2 \\ A_3 & \mathbf{0} \\ \mathbf{0} & \hat{A}_4 \end{bmatrix} \in \mathbb{R}^{(M+1) \times N} \text{ with}$$

$$\begin{cases} A_1 = I_m \otimes \mathbf{1}_{d_1}^\top \in \mathbb{R}^{m \times d_1^2 d_2}, \\ A_2 = -\mathbf{1}_{d_2}^\top \otimes I_m \in \mathbb{R}^{m \times d_1 d_2^2}, \\ A_3 = I_{d_2} \otimes (\mathbf{1}_{d_1}^\top \otimes I_{d_1}) \in \mathbb{R}^{m \times d_1^2 d_2}, \\ \hat{A}_4 = I_{d_2} \otimes (\mathbf{1}_{d_2}^\top \otimes I_{d_1}) \in \mathbb{R}^{m \times d_1 d_2^2}. \end{cases} \quad (4.82)$$

For notational convenience, let $\bar{I}_{d_1} = [I_{d_1-1}, \mathbf{0}_{d_1-1}] \in \mathbb{R}^{(d_1-1) \times d_1}$. We define

$$A := \begin{bmatrix} A_1 & A_2 \\ A_3 & \mathbf{0} \\ \mathbf{0} & A_4 \end{bmatrix} \in \mathbb{R}^{M \times N}, \quad b := [0_m; \mu^1; \bar{I}_m \mu^2] \in \mathbb{R}^M. \quad (4.83)$$

with

$$A_4 = \text{diag}(\mathbf{1}_{d_2}^\top \otimes I_m, \dots, \mathbf{1}_{d_2}^\top \otimes I_m, \mathbf{1}_{d_2}^\top \otimes \bar{I}_{d_1}) \in \mathbb{R}^{(m-1) \times d_1 d_2^2}. \quad (4.84)$$

Similar to (Dantzig and Thapa, 2003, Lemma 7.1), we can obtain that A defined in (4.83) has full row rank, and

$$\{x \in \mathbb{R}^N \mid Ax = b\} = \{x \in \mathbb{R}^N \mid \hat{A}x = \hat{b}\}.$$

As a result, the LP problem (4.81) is equivalent to

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & \langle c, x \rangle + \delta_{\mathbb{R}_+^N}(x) \\ \text{s.t.} \quad & Ax = b. \end{aligned} \quad (4.85)$$

Furthermore, the dual problem of (4.85) takes the form:

$$\min_{y \in \mathbb{R}^M, z \in \mathbb{R}^N} \left\{ -\langle b, y \rangle + \delta_{\mathbb{R}_+^N}(z) \mid A^*y + z = c \right\}. \quad (4.86)$$

The KKT conditions associated with (4.85) and (4.86) can be given by

$$A^*y + z = c, \quad Ax = b, \quad \mathbb{R}_+^N \ni x \perp z \in \mathbb{R}_+^N, \quad (4.87)$$

where $x \perp z$ means x is perpendicular to z , i.e., $\langle x, z \rangle = 0$.

4.3.2 Reconstruct the Transport Plan from the Reduced Model

The absence of the transport plan π makes the reduced OT model less favorable in applications where the transport map is necessary (i.e., color transfer (Pitié and Kokaram, 2007; Bonneel et al., 2013; Solomon et al., 2015)). We address this issue by proposing a fast algorithm (shown in Algorithm 9) to reconstruct an OT plan of the original model from an optimal solution of the reduced model (4.80).

Algorithm 9 A fast algorithm for reconstructing transport plan π from the network flows $f^{(1)}$ and $f^{(2)}$.

Input: An optimal flow $(f^{(1)}, f^{(2)})$ of problem (4.80).

Output: An OT mapping π of problem (4.78).

for $(k, j) \in \mathcal{I}$ **do**

for $i = 1, \dots, d_1$ **do**

for $l = 1, \dots, d_2$ **do**

$$\pi_{i,j;k,l} = \min\{f_{i,k,j}^{(1)}, f_{k,j,l}^{(2)}\}$$

$$f_{i,k,j}^{(1)} = f_{i,k,j}^{(1)} - \pi_{i,j;k,l}$$

$$f_{k,j,l}^{(2)} = f_{k,j,l}^{(2)} - \pi_{i,j;k,l}$$

The following proposition shows that the output of Algorithm 9 is an OT plan for the original OT model.

Proposition 4.4. *Given an optimal solution $(f^{(1)}, f^{(2)})$ to problem (4.80), the output π of Algorithm 9 is an optimal solution to the OT problem (4.78).*

Proof. Since $(f^{(1)}, f^{(2)})$ is an optimal solution to problem (4.80), we have

$$f_{i,k,j}^{(1)} \geq 0, \quad f_{k,j,l}^{(2)} \geq 0 \quad \forall (i, j) \in \mathcal{I} \quad \forall (k, l) \in \mathcal{I},$$

which implies

$$\pi_{i,j;k,l} \geq 0 \quad \forall (i, j) \quad \forall (k, l) \in \mathcal{I}. \quad (4.88)$$

Fix i, j, k . For convenience, let $f_{i,k,j}^{(1)-l}$ denote the value of $f_{i,k,j}^{(1)}$ at the l -th iteration before updating. According to the update formula of $f_{i,k,j}^{(1)}$, there must exist an index

$1 \leq l^* \leq d_2$ such that $f_{i,k,j}^{(1)-l^*} \leq f_{k,j,l^*}^{(2)}$. Otherwise, we would have

$$f_{i,k,j}^{(1)} > \sum_{l=1}^{d_2} f_{k,j,l}^{(2)},$$

which contradicts the feasibility of $(f^{(1)}, f^{(2)})$:

$$f_{i,k,j}^{(1)} \leq \sum_{i=1}^{d_1} f_{i,k,j}^{(1)} = \sum_{l=1}^{d_2} f_{k,j,l}^{(2)}.$$

Therefore, after d_2 iterations, the final value of $f_{i,k,j}^{(1)}$ must be zero, which leads to

$$\sum_{l=1}^{d_2} \pi_{i,j;k,l} = f_{i,k,j}^{(1)}. \quad (4.89)$$

Similarly, by fixing j, k, l , we also obtain

$$\sum_{i=1}^{d_1} \pi_{i,j;k,l} = f_{k,j,l}^{(2)}. \quad (4.90)$$

Hence, according to the constraints in problem (4.80), we have

$$\begin{cases} \sum_{(k,l) \in \mathcal{I}} \pi_{i,j;k,l} = \sum_{k=1}^{d_1} f_{i,k,j}^{(1)} = \mu_{i,j}^1, \\ \sum_{(i,j) \in \mathcal{I}} \pi_{i,j;k,l} = \sum_{j=1}^{d_2} f_{k,j,l}^{(2)} = \mu_{k,l}^2, \end{cases}$$

which, together with (4.88), shows that π is a feasible solution to problem (4.78).

Furthermore, from (4.89), (4.90), and the definition of c in (4.79), we can obtain

$$\begin{aligned} & \sum_{((i,j),(k,l))} c_{i,j;k,l} \pi_{i,j;k,l} \\ &= \sum_{(i,j) \in \mathcal{I}} \left[\sum_{k=1}^{d_1} (k-i)^2 f_{i,k,j}^{(1)} + \sum_{l=1}^{d_2} (j-l)^2 f_{k,j,l}^{(2)} \right]. \end{aligned}$$

According to (Auricchio et al., 2018, Theorem 1), we know that the optimal objective function values of problems (4.78) and (4.80) are equivalent. Therefore, π is an optimal solution to problem (4.78). $\square$

4.3.3 HOT: A Halpern Accelerating Method for Solving OT Problems

In this section, we first introduce an efficient Halpern accelerating method for solving problem (4.86), which includes the equivalent reduced OT problem (4.85) as a special case. Subsequently, we present an efficient implementation of the proposed algorithm by designing a novel procedure to solve the involved linear system in linear time complexity. Given $\sigma > 0$, the augmented Lagrange function corresponding to the dual problem (4.86) is defined by, for any $(y, z, x) \in \mathbb{R}^M \times \mathbb{R}^N \times \mathbb{R}^N$,

$$L_\sigma(y, z; x) := -\langle b, y \rangle + \delta_{\mathbb{R}_+^N}(z) + \frac{\sigma}{2} \|A^*y + z - c + \frac{1}{\sigma}x\|^2 - \frac{1}{2\sigma} \|x\|^2.$$

For ease of notation, denote $w := (y, z, x)$. An accelerated pADMM with $\alpha = 2$ in Algorithm 8 for solving OT problems is presented in Algorithm 10.

Algorithm 10 HOT: A Halpern accelerating method for solving the OT problem (4.86)

- 1: **Input:** Choose $w^0 = (y^0, z^0, x^0) \in \mathbb{R}^M \times \mathbb{R}^N \times \mathbb{R}^N$, and set $\sigma > 0$. Denote $\bar{w} = (\bar{y}, \bar{z}, \bar{x})$.
 - 2: **for** $k = 0, 1, \dots$ **do**
 - 3: Step 1. $\bar{y}^k = \arg \min_{y \in \mathbb{R}^M} L_\sigma(y, z^k; x^k)$
 - 4: Step 2. $\bar{x}^k = x^k + \sigma(A^*\bar{y}^k + z^k - c)$
 - 5: Step 3. $\bar{z}^k = \arg \min_{z \in \mathbb{R}^N} L_\sigma(\bar{y}^k, z; \bar{x}^k)$
 - 6: Step 4. $w^{k+1} = \frac{1}{k+2} w^0 + \frac{k+1}{k+2} (2\bar{w}^k - w^k)$
-

Proposition 4.5. *The sequence $\{\bar{w}^k\} = \{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ generated by the HOT algorithm in Algorithm 10 converges to the point $w^* = (y^*, z^*, x^*)$, where (y^*, z^*) is a solution to problem (4.86) and x^* is a solution to problem (4.85).*

Proof. The result follows from Corollary 4.3. □

Next, we analyze the iteration complexity of the HOT algorithm for obtaining an ε -approximate solution, where an appropriate measure for the quality of the solution

is crucial. In this thesis, we consider the residual mapping associated with the KKT system (4.87):

$$\mathcal{R}(w) = \begin{pmatrix} b - Ax \\ z - \Pi_{\mathbb{R}_+^N}(z - x) \\ c - A^*y - z \end{pmatrix}$$

for any $w = (y, z, x) \in \mathbb{R}^M \times \mathbb{R}^N \times \mathbb{R}^N$. Note that $\mathcal{R}(w^*) = 0$ is equivalent to the facts that $x^* \in \mathbb{R}^N$ and $(y^*, z^*) \in \mathbb{R}^M \times \mathbb{R}^N$ are the solution to problems (4.85) and (4.86), respectively. The KKT residual $\|\mathcal{R}(\cdot)\|$ is a commonly used and practical measure for the quality of the approximation solution to (4.85).

Proposition 4.6. *Let $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ be the sequence generated by Algorithm 10, and let $w^* = (y^*, z^*, x^*)$ be the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k, \bar{x}^k)\}$ and $R_0 = \|x^0 - x^* + \sigma(z^0 - z^*)\|$. For all $k \geq 0$, we have the following bound:*

$$\|\mathcal{R}(\bar{w}^k)\| \leq \left(\frac{\sigma + 1}{\sigma} \right) \frac{R_0}{(k + 1)}. \quad (4.91)$$

Proof. The result follows from Theorem 4.2. □

Remark 4.6. *Note that, without acceleration, the ADMM has an $O(1/\sqrt{k})$ non-ergodic rate in terms of both the objective function value gap and feasibility violations (Davis and Yin, 2016; Cui et al., 2016). In contrast, the HOT algorithm in Algorithm 10 achieves an $O(1/k)$ non-ergodic convergence rate, offering significant advantages for solving large-scale OT problems.*

Now, we present a fast implementation of the HOT algorithm. Through direct calculations, we obtain the following updates of $\bar{z}^k$ and $\bar{y}^k$ for any $k \geq 0$:

1. Update of $\bar{z}^k$:

$$\bar{z}^k = \Pi_{\mathbb{R}_+^N} (c - A^*\bar{y}^k - \bar{x}^k/\sigma);$$

2. Update of $\bar{y}^k$:

$$AA^*\bar{y}^k = \frac{b}{\sigma} - A \left(\frac{x^k}{\sigma} + z^k - c \right). \quad (4.92)$$

Therefore, the main computational bottleneck of the HOT algorithm for solving (4.85) is solving the linear system (4.92). Note that the dimension of the matrix AA^* is $M \times M$ with M defined in (4.81). In applications, M is usually a huge number. As an illustrative example, for an image with 256×256 pixels, $M = 196,607$. As a result, it is not computationally affordable for computing a (sparse) Cholesky decomposition for the matrix AA^* or solving the linear system (4.92) with a standard direct solver. Indeed, it is computationally expensive even for computing the matrix AA^* . Instead, in the remaining part of this subsection, we will derive a linear time complexity procedure for solving the linear equation $AA^*y = R$ with a given vector $R \in \mathbb{R}^M$. It is worthwhile mentioning that our procedure does not require calculating or storing the matrix AA^* . Note that AA^* can be written in the following form:

$$AA^* = \begin{bmatrix} E_1 & E_2 & E_3 \\ E_2^* & E_4 & \mathbf{0} \\ E_3^* & \mathbf{0} & E_5 \end{bmatrix}, \quad (4.93)$$

where

1. $E_1 = (d_1 + d_2)I_m \in \mathbb{R}^{m \times m}$;
2. $E_2 = \text{diag}(\mathbf{1}_{d_1}\mathbf{1}_{d_1}^\top, \dots, \mathbf{1}_{d_1}\mathbf{1}_{d_1}^\top, \mathbf{1}_{d_1}\mathbf{1}_{d_1}^\top) \in \mathbb{R}^{m \times m}$;
3. $E_3 = -\mathbf{1}_{d_2} \otimes (I_m, \dots, I_m, \bar{I}_{d_1}^*) \in \mathbb{R}^{m \times (m-1)}$;
4. $E_4 = d_1 I_m \in \mathbb{R}^{m \times m}$;
5. $E_5 = A_4 A_4^* = d_2 I_{m-1} \in \mathbb{R}^{(m-1) \times (m-1)}$.

To better explore the structure of the linear system $AA^*y = R$, we rewrite it equivalently as

$$AA^*y = \begin{bmatrix} E_1 & E_2 & E_3 \\ E_2^* & E_4 & \mathbf{0} \\ E_3^* & \mathbf{0} & E_5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}, \quad (4.94)$$

where $y := (y_1; y_2; y_3) \in \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^{m-1}$ and $R := (R_1; R_2; R_3) \in \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^{m-1}$.

To further explore the block structure of the linear system, we can denote $y_i := (y_i^1; \dots; y_i^{d_2}) \in \mathbb{R}^{d_1} \times \dots \times \mathbb{R}^{d_1}$ for $i = 1, 2$, and $y_3 = (y_3^1; \dots; y_3^{d_2}) \in \mathbb{R}^{d_1} \times \dots \times \mathbb{R}^{d_1} \times \mathbb{R}^{d_1-1}$. Correspondingly, we write $R_i = (R_i^1; \dots; R_i^{d_2})$ for $i = 1, 2, 3$. The next proposition gives an explicit formula of the solution to the linear equation in (4.94).

Proposition 4.7. *Consider $A \in \mathbb{R}^{M \times N}$ defined in (4.83). Given $R \in \mathbb{R}^M$, the solution y to $AA^*y = R$ in the form (4.94) is given by:*

$$y_2^j = \frac{1}{d_1} (R_2^j - \mathbf{1}_{d_1}^\top y_1^j \mathbf{1}_{d_1}) \quad j = 1, \dots, d_2, \quad (4.95)$$

$$y_3^j = \frac{1}{d_2} (R_3^j + \sum_{s=1}^{d_2} y_1^s) \quad j = 1, \dots, d_2 - 1, \quad (4.96)$$

$$y_3^{d_2} = \frac{1}{d_2} (R_3^{d_2} + \bar{I}_{d_1} \sum_{j=1}^{d_2} y_1^j), \quad (4.97)$$

$$y_1^j = \hat{y}_1^j - \hat{y}_1^a \quad j = 1, \dots, d_2, \quad (4.98)$$

where

1. $\hat{y}_1^j = \frac{1}{d_1 + d_2} (\tilde{R}_1^j + \tilde{R}_2^j + \tilde{R}_3), \quad j = 1, \dots, d_2$, with $\tilde{R}_1^j = R_1^j + \frac{1}{d_2} \mathbf{1}_{d_1}^\top R_1^j \mathbf{1}_{d_1}$, $\tilde{R}_2^j = -\left(\frac{1}{d_1} + \frac{1}{d_2}\right) \mathbf{1}_{d_1}^\top R_2^j \mathbf{1}_{d_1}$, and $\tilde{R}_3 = \frac{1}{d_2} \left(\sum_{j=1}^{d_2-1} R_3^j + \bar{I}_{d_1}^* R_3^{d_2} \right) + \frac{1}{d_2^2} \mathbf{1}_{m-1}^\top R_3 \mathbf{1}_{m-1}$;
2. $\hat{y}_1^a = \left(I_{d_1} + \frac{1}{d_2} \mathbf{1}_{d_1} \mathbf{1}_{d_1}^\top \right) \hat{W} \sum_{j=1}^{d_2} \hat{y}_1^j$;

$$\begin{aligned} 3. \quad \hat{W} &= \left(-\text{diag} \left(\frac{1}{d_1} I_{d_1-1}, \frac{1}{d_1+1} \left(1 - \frac{1}{d_2} \right) \right) - \frac{1}{w} dd^\top \right), \text{ with } d = \left[\frac{1}{d_1} \mathbf{1}_{d_1-1}; \frac{1}{d_1+1} \left(1 - \frac{1}{d_2} \right) \right] \in \\ &\mathbb{R}^{d_1} \text{ and } w = \frac{1}{d_1} - \frac{1}{(d_1+1)} \left(1 - \frac{1}{d_2} \right). \end{aligned}$$

Proof. By some direct calculations, we can solve (4.94) equivalently as:

$$y_2 = \frac{R_2 - E_2^* y_1}{d_1}, \quad (4.99)$$

$$y_3 = \frac{R_3 - E_3^* y_1}{d_2}, \quad (4.100)$$

$$\tilde{E}_1 y_1 = R_1 - \frac{1}{d_1} E_2 R_2 - \frac{1}{d_2} E_3 R_3, \quad (4.101)$$

where $\tilde{E}_1 = \left(E_1 - \frac{1}{d_1} E_2 E_2^* - \frac{1}{d_2} E_3 E_3^* \right)$. As a result, the key is to obtain y_1 by solving (4.101). Let $\hat{E}_1 := E_1 - \frac{1}{d_1} E_2 E_2^*$. By direct calculations, we have:

$$\hat{E}_1 = \text{diag} \left(\hat{E}_1^1, \dots, \hat{E}_1^{d_2} \right) \quad (4.102)$$

with $\hat{E}_1^j = (d_1 + d_2) I_{d_1} - \mathbf{1}_{d_1} \mathbf{1}_{d_1}^\top$, $j = 1, \dots, d_2$. By the Sherman–Morrison–Woodbury formula, we directly get:

$$\hat{E}_1^{-1} = \text{diag} \left((\hat{E}_1^1)^{-1}, \dots, (\hat{E}_1^{d_2})^{-1} \right) \quad (4.103)$$

with $(\hat{E}_1^j)^{-1} = \frac{1}{d_1 + d_2} \left(I_{d_1} + \frac{1}{d_2} \mathbf{1}_{d_1} \mathbf{1}_{d_1}^\top \right)$, $j = 1, \dots, d_2$. On the other hand, let:

$$Q = \frac{d_2 - 1}{d_2} I_{d_1} + \frac{1}{d_2} \bar{I}_{d_1}^* \bar{I}_{d_1} \in \mathbb{R}^{d_1 \times d_1}.$$

Denote $\hat{Q} := Q^{1/2}$ such that $\hat{Q} \hat{Q} = Q$, and $\bar{Q} := \mathbf{1}_{d_2} \otimes \hat{Q}$. We can obtain:

$$\tilde{E}_1 = \left(\hat{E}_1 - \frac{1}{d_2} E_3 E_3^* \right) = \hat{E}_1 - \bar{Q} \bar{Q}^*.$$

Hence, by the Sherman–Morrison–Woodbury formula, we can derive:

$$\begin{aligned}\tilde{E}_1^{-1} &= \left(\hat{E}_1 - \frac{1}{d_2} E_3 E_3^* \right)^{-1} \\ &= \hat{E}_1^{-1} - \hat{E}_1^{-1} \bar{Q} W^{-1} \bar{Q}^* \hat{E}_1^{-1}\end{aligned}\tag{4.104}$$

with $W = -I_m + \bar{Q}^* \hat{E}_1^{-1} \bar{Q}$. Note that:

$$\begin{aligned}W &= -I_m + \sum_{j=1}^{d_2} \hat{Q}(E_1^j)^{-1} \hat{Q} \\ &= \frac{1}{d_1+d_2} \left(\text{diag}(-d_1 I_{d_1-1}, -(d_1+1)) + d_0 d_0^\top \right),\end{aligned}$$

where $d_0 = [\mathbf{1}_{d_1-1}; \sqrt{1 - \frac{1}{d_2}}] \in \mathbb{R}^{d_1}$. Applying the Sherman–Morrison–Woodbury formula to W , we can obtain:

$$W^{-1} = (d_1 + d_2) \left(\text{diag}\left(-\frac{1}{d_1} I_{d_1-1}, -\frac{1}{d_1+1}\right) - \frac{1}{d_0} \hat{d}_0 \hat{d}_0^\top \right)$$

with $\hat{d}_0 = [\frac{1}{d_1} \mathbf{1}_{d_1-1}; \frac{1}{d_1+1} \sqrt{1 - \frac{1}{d_2}}]$. It follows that

$$\begin{aligned}\bar{Q} W^{-1} \bar{Q}^* &= \mathbf{1}_{d_2} \mathbf{1}_{d_2}^\top \otimes (\hat{Q} W^{-1} \hat{Q}) \\ &= \mathbf{1}_{d_2} \mathbf{1}_{d_2}^\top \otimes (d_1 + d_2) \hat{W}.\end{aligned}\tag{4.105}$$

To explore the block structure of R , we denote:

$$\hat{R}_1 = R_1 - \frac{1}{d_1} E_2 R_2 - \frac{1}{d_2} E_3 R_3 = (\hat{R}_1^1; \dots; \hat{R}_1^{d_2}) \in \mathbb{R}^{d_1} \times \dots \times \mathbb{R}^{d_1},$$

which implies

$$\hat{R}_1^j = R_1^j - \frac{1}{d_1} \mathbf{1}_{d_1}^\top R_2^j + \frac{1}{d_2} \left(\sum_{j=1}^{d_2-1} R_3^j + \bar{I}_{d_1}^* R_3^{d_2} \right), \quad j = 1, \dots, d_2.$$

Then, for $j = 1, \dots, d_2$,

$$\begin{aligned}\hat{y}_1^j &= (\hat{E}_1^{-1} \hat{R}_1)^j \\ &= \frac{1}{d_1+d_2} \left(\tilde{R}_1^j + \tilde{R}_2^j + \tilde{R}_3 \right).\end{aligned}$$

Define

$$\hat{y}_1^a := \left(I_{d_1} + \frac{1}{d_2} \mathbf{1}_{d_1} \mathbf{1}_{d_1}^\top \right) \hat{W} \sum_{j=1}^{d_2} \hat{y}_1^j.$$

From (4.101), (4.104), and (4.105), we have:

$$y_1^j = \hat{y}_1^j - \hat{y}_1^a, \quad j = 1, \dots, d_2.$$

Substituting y_1 into (4.99) and (4.100), we can obtain the results for y_2 and y_3 . This completes the proof. $\square$

According to the explicit formula in Proposition 4.7, we can immediately derive the complexity result for solving the linear equation in (4.94).

Corollary 4.4. *Consider $A \in \mathbb{R}^{M \times N}$ defined in (4.83). The linear system $AA^*y = R$ in the form (4.94) can be solved in $O(M)$ flops.*

Based on this corollary, we can determine the per-iteration computational cost of the HOT algorithm in Algorithm 10 for each iteration.

Corollary 4.5. *The per-iteration computational complexity of the HOT algorithm in Algorithm 10 in terms of flops is $O(N)$.*

Proof. Since A has at most $2N$ nonzero elements, the computational cost for calculating Ax and A^*y is only $O(N)$. Except for solving linear systems, which can be done in $O(M)$ from Corollary 4.4, Algorithm 10 primarily involves matrix-vector multiplications and vector additions. Hence, the computational cost of Algorithm 10 for each iteration is $O(N)$. $\square$

Remark 4.7. *Leveraging the sparsity of A , we can compute Ax and A^*y with linear space complexity $O(N)$. Furthermore, according to Proposition 4.7, the linear system $AA^*y = R$ can be solved with a memory cost of $O(M)$. Hence, Algorithm 10 can be*

implemented with linear space complexity $O(N)$. When $d_1 = d_2$, this corresponds to $O(m^{1.5})$.

Combining the iteration complexity in Proposition 4.6 and the computational cost for each iteration in Corollary 4.5, we can derive the following overall computational complexity result of the HOT algorithm for solving problem (4.80).

Theorem 4.3. *Let $\{\bar{y}^k, \bar{z}^k, \bar{x}^k\}$ be the sequence generated by the HOT algorithm in Algorithm 10. For any given tolerance $\varepsilon > 0$, the HOT algorithm needs at most*

$$\frac{1}{\varepsilon} \left(\frac{1 + \sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) - 1$$

iterations to return a solution to the equivalent OT problem (4.85) such that the KKT residual $\|\mathcal{R}(\bar{w}^k)\| \leq \varepsilon$, where (x^*, z^*) is the limit point of the sequence $\{\bar{x}^k, \bar{z}^k\}$. In particular, the overall computational complexity of the HOT algorithm in Algorithm 10 to achieve this accuracy in terms of flops is

$$O \left(\left(\frac{1 + \sigma}{\sigma} (\|x^0 - x^*\| + \sigma \|z^0 - z^*\|) \right) \frac{d_1^2 d_2 + d_1 d_2^2}{\varepsilon} \right).$$

Remark 4.8. *Note that the complexity of the Sinkhorn method for solving the original OT problem (4.78) is $O\left(\frac{d_1^2 d_2^2}{\varepsilon^2}\right)$ to achieve an ε -accuracy solution in terms of objective function value (Dvurechensky et al., 2018). Even by leveraging the separable structure of the cost function c defined in (4.79), the convolutional Sinkhorn method, as mentioned in (Solomon et al., 2015) and (Auricchio et al., 2018), still requires $O\left(\frac{d_1^2 d_2 + d_1 d_2^2}{\varepsilon^2} + d_1^2 d_2^2\right)$ to compute the KW distance. In contrast, the HOT algorithm exhibits lower overall computational complexity, offering a significant advantage in calculating the KW distance between large-scale histograms.*

Remark 4.9. *Assume that $d_1 = d_2$. Blanchet et al. (Blanchet et al., 2024) demonstrated that a running time of $\tilde{O}(m^2/\varepsilon)$ is a bottleneck in their approach by reducing*

an instance of the bipartite matching problem (BMP) to an OT problem. While HOTA achieves a complexity of $O(m^{3/2}/\varepsilon)$, it cannot be directly applied to solve the maximum cardinality BMP. This is because, in the reduction from the BMP to the OT problem, the ground distance used in (Blanchet et al., 2024) differs from the one in our paper. Consequently, the hardness result in (Blanchet et al., 2024) does not directly contradict our complexity bound.

4.4 Experiments

In this section, we comprehensively compare the HOTA algorithm with five other state-of-the-art methods on the popular DOTmark dataset (Schrieber et al., 2016). Additionally, employing the transport plan derived from Algorithm 9, we test the performance of the reduced OT model on the color transfer task. The numerical results reveal that HOTA provides significant advantages over state-of-the-art methods in both memory and computational efficiency, particularly for large-scale problems.

4.4.1 Numerical Comparison on the DOTmark Dataset

The DOTmark dataset (Schrieber et al., 2016) is a comprehensive collection of benchmark instances for evaluating and comparing algorithms in the field of OT. It consists of a variety of instances categorized into different classes, such as Classic Images, Shapes, and Gaussian Distributions. In this experiment, we selected eight images each from the Classic Images and Shapes categories, as illustrated in Figure 4.1. These selected images were resized to four different resolutions: 64×64 , 128×128 , 256×256 , and 512×512 . Finally, we randomly selected 10 pairs from each category and computed the KW distance for each pair.

To exhibit the superiority of HOTA, we compare it with the following five state-of-the-art methods:

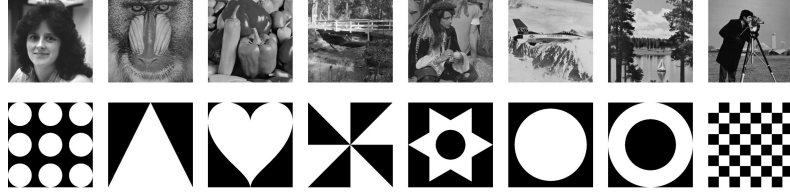


Figure 4.1: A visualization of the selected images from the DOTmark Dataset is presented. The upper row features images from the Classic Images category, while the bottom row contains images from the Shapes category.

- **Sinkhorn** (Cuturi, 2013) is a widely used algorithm for computing an approximate KW distance by solving an entropy-regularized OT problem. Since its performance is sensitive to the regularization parameter, denoted as λ . To achieve varying levels of solution accuracy, we selected the λ to be 1%, 0.1%, and 0.01% of the median transport cost, following the setup from (Solomon et al., 2015). Due to the potential numerical instability caused by small λ values, we employed the Log-domain Sinkhorn algorithm implemented by POT (Flamary et al., 2021) as the baseline method.
- **Convolutional Sinkhorn** (Solomon et al., 2015; Solomon, 2018) is an improved version of the Sinkhorn algorithm, optimized for computing distances over regular two-dimensional grids. Specifically, since the kernel matrix used by the Sinkhorn algorithm can be constructed using a Kronecker product of two smaller matrices, a matrix-vector product using a matrix of dimension $m \times m$ can be replaced by two matrix-matrix products over matrices of dimension $\sqrt{m} \times \sqrt{m}$, resulting in a significant improvement in computational efficiency. In our experiment, based on the MATLAB implementation of the convolutional Sinkhorn in (Auricchio et al., 2018), we further developed the Log-domain convolutional Sinkhorn using PyTorch. The regularization parameter λ is kept the same as that of the original Sinkhorn method.
- **Gurobi (11.0.1)** is a popular optimization solver designed to address a wide

range of mathematical programming problems, including linear and quadratic programming. In our experiment, we employ the interior point method (IPM) implemented in Gurobi to solve the reduced model (1.1). Since it is unnecessary to obtain a basic solution, we disable the crossover strategy.

- **Network Simplex** implemented in the Lemon C++ graph library ¹, is a highly efficient algorithm for solving uncapacitated minimum cost flow problems (Kovács, 2015). It has demonstrated the computational advantages over the Sinkhorn-type methods in (Auricchio et al., 2018) for small size images by solving the reduced model (1.1).
- **ADMM** (Gabay and Mercier, 1976; Glowinski and Marroco, 1975) is a popular first-order primal-dual method for solving large-scale optimization problems. It has shown great potential in solving large-scale OT problems (4.78) in a GPU setting (Mai et al., 2022). In this numerical experiment, we use a generalized ADMM (Eckstein and Bertsekas, 1992; Xiao et al., 2018) to solve the equivalent model (1.1) (replacing Step 4 in Algorithm 10 with $w^{k+1} = (1 - \rho)w^k + \rho\bar{w}^k$ and setting $\rho = 1.7$) as the baseline to evaluate the acceleration effect of the Halpern iteration.

All experiments are conducted on an Ubuntu 22.04 server equipped with an Intel(R) Xeon(R) Platinum 8480C processor and an Nvidia GeForce RTX 4090 GPU with 24 GB of RAM. Due to hardware specifications, we limit the maximum memory usage of each algorithm to 24 GB. Additionally, we set the maximum one-call running time of each algorithm to 3600 seconds. For the HOT and ADMM methods, we adopt a stopping criterion based on the relative KKT residual:

$$\text{KKT}_{\text{res}} = \max \left\{ \frac{\|A^*y + z - c\|}{1 + \|c\|}, \frac{\|\min(x, z)\|}{1 + \|x\| + \|z\|}, \frac{\|Ax - b\|}{1 + \|b\|} \right\}. \quad (4.106)$$

¹ <https://lemon.cs.elte.hu/>

Other methods use their default stopping criteria. We terminate all tested algorithms, except for Network Simplex which is an exact algorithm with a stopping tolerance of 10^{-6} . Finally, since different methods have varying stopping criteria, we use the following metrics to fairly evaluate the quality of the solutions: the ‘relative objective gap’ (gap) and the ‘relative primal feasibility error’ (feaserr). These metrics are defined as follows:

$$\text{gap} = \frac{|\langle c, x \rangle - \langle c, x_b \rangle|}{|\langle c, x_b \rangle| + 1},$$

$$\text{feaserr} = \max \left\{ \frac{\|\min(x, \mathbf{0})\|}{1 + \|x\|}, \frac{\|Ax - b\|}{1 + \|b\|} \right\},$$

where x_b is the solution obtained using Gurobi with the tolerance set to 10^{-8} .

We present the average results of 10 pairs within each category for all tested algorithms in Table 4.1. Since Gurobi runs out of memory for images sized from 256×256 to 512×512 , we only report the feaserr for the HOT, Network Simplex, and ADMM. Additionally, Table 4.1 only shows the results of Sinkhorn-type methods with $\lambda = 0.01\%$ of the median transport cost, as this parameter returns a solution of comparable quality to other methods. For more results of the convolutional Sinkhorn with different regularization parameters, refer to Table 4.2. Due to space constraints, we omit the results of the Sinkhorn with varying parameters, which exhibit similar performance to the convolutional Sinkhorn in terms of the gap.

Table 4.1: The numerical results of different algorithms on the DOTmark dataset.

Category	Resolution		HOT	Network Simplex	Gurobi	ADMM	Convolutional Sinkhorn	Sinkhorn
Classic	64×64	time(s)	0.67	2.73	2.16	1.77	16.18	174.82
		gap	8.26E-04	3.46E-10	1.20E-04	2.67E-04	1.69E-04	2.12E-04
		feaserr	4.58E-07	4.88E-32	2.55E-11	3.09E-07	7.90E-07	9.75E-07
		iter	1700	-	13	3420	64126	62474
	128×128	time(s)	1.58	36.18	29.15	3.53	39.40	2632.17
		gap	6.24E-03	8.74E-10	1.07E-04	1.72E-03	6.98E-04	6.35E-04
		feaserr	7.27E-07	9.67E-32	7.24E-12	3.73E-07	8.34E-07	9.87E-07
		iter	1170	-	14	3240	58446	57010
	256×256	time(s)	12.98	2562.92		20.80		
		feaserr	8.05E-07	1.35E-31	Memory Overflow	6.04E-07	Memory Overflow	Memory Overflow
		iter	1140	-		2250		
	512×512	time(s)	81.02	Over Maximum Running Time		116.92		
		feaserr	3.28E-07		Memory Overflow	4.32E-07	Memory Overflow	Memory Overflow
		iter	900			1610		
Shapes	64×64	time(s)	0.64	1.48	1.33	3.92	9.60	103.74
		gap	3.78E-04	1.81E-10	2.28E-05	5.85E-05	4.86E-05	6.07E-05
		feaserr	5.77E-07	7.24E-32	1.88E-10	2.58E-07	7.95E-07	9.68E-07
		iter	1610	-	15	10430	37986	37077
	128×128	time(s)	1.68	20.70	22.46	2.32	24.32	1616.34
		gap	2.51E-03	2.46E-09	2.19E-05	4.11E-04	3.28E-04	3.09E-04
		feaserr	1.01E-06	1.16E-31	2.01E-10	7.74E-07	8.01E-07	9.83E-07
		iter	1240	-	18	2130	36080	35009
	256×256	time(s)	14.87	959.77		23.30		
		feaserr	6.68E-07	1.59E-31	Memory Overflow	7.17E-07	Memory Overflow	Memory Overflow
		iter	1310	-		2530		
	512×512	time(s)	87.12	Over Maximum Running Time		118.10		
		feaserr	3.54E-07		Memory Overflow	5.71E-07	Memory Overflow	Memory Overflow
		iter	970			1630		

Table 4.2: The numerical results of the convolutional Sinkhorn method with different λ .

Solver	Category	Resolution		$\lambda = 0.01\%$	$\lambda = 0.1\%$	$\lambda = 1\%$
Convolutional Sinkhorn	Classic	64×64	time(s)	16.18	1.60	0.19
			gap	1.69E-04	2.43E-02	3.29E-01
			feaserr	7.90E-07	8.15E-07	7.10E-07
			iter	64126	6400	650
		128×128	time(s)	39.40	3.94	0.39
			gap	6.98E-04	3.34E-02	3.51E-01
			feaserr	8.34E-07	8.29E-07	7.29E-07
			iter	58446	5850	594

We summarized some key findings in Table 4.1 from the perspective of computational efficiency and memory cost:

1. **Computational efficiency:** HOT can return a comparable solution in terms of the gap and feaserr in the shortest time. Although the Network Simplex and Gurobi have computational advantages for computing the KW distance for small-scale images, these methods cannot handle large-scale problems effectively. IPM implemented in Gurobi requires solving a linear equation that relies heavily on Cholesky decomposition, causing the computational cost of each iteration to increase rapidly with image size. Additionally, the inherent sequential nature of the Network Simplex algorithm makes it challenging to parallelize effectively. In contrast, HOT and convolutional Sinkhorn benefit from low per-iteration costs and are easily parallelizable. However, the convolutional Sinkhorn needs to recover the solution to the original problem (4.78) to compute the KW distance, making it unsuitable for large-scale problems. As a result, for images sized 128×128 , HOT achieves a 17.44x speedup over Network Simplex, a 15.83x speedup over Gurobi, and a 19.54x speedup over

convolutional Sinkhorn. Additionally, compared to ADMM, HOT benefits from a superior $O(1/k)$ iteration complexity and saves 40% of iterations for images sized 512×512 .

2. **Memory cost:** HOT demonstrates a significant advantage in memory efficiency by exploiting the sparse structure of A in (4.82) and utilizing the explicit solution of the linear equation presented in Proposition 4.7 to avoid sparse Cholesky decomposition. In contrast, Sinkhorn-type methods need to recover the solution to the original OT problem (4.78) and maintain the transport cost to calculate the KW distance, which involves m^2 variables. Given that these variables are stored using 64-bit floating-point representation, the Sinkhorn-based methods require at least 32 GB of memory for images sized 256×256 , which far exceeds the available 24 GB. While the IPM in Gurobi solves the reduced model (1.1), each iteration requires performing a sparse Cholesky decomposition, making it unsuitable for images sized 256×256 or larger.

To further illustrate the benefit of the explicit solution of the linear system presented in Proposition 4.7, we conduct a comparison between solving the linear equation using Proposition 4.7 and sparse Cholesky decomposition. Given that the average iteration number of HOT is around 1500, we solve the linear system (4.94) 1500 times for different image sizes. The results are shown in Figure 4.2. It is clear that the computational costs of Cholesky decomposition increase rapidly as image size increases. Additionally, the forward-backward substitution method used to solve the linear system is also time-consuming because it cannot be efficiently parallelized. In contrast, solving linear systems using Proposition 4.7 can be easily parallelized. It consumes significantly less time and remains constant as the size varies because it only requires $O(M)$ flops, as shown in Corollary 4.4.

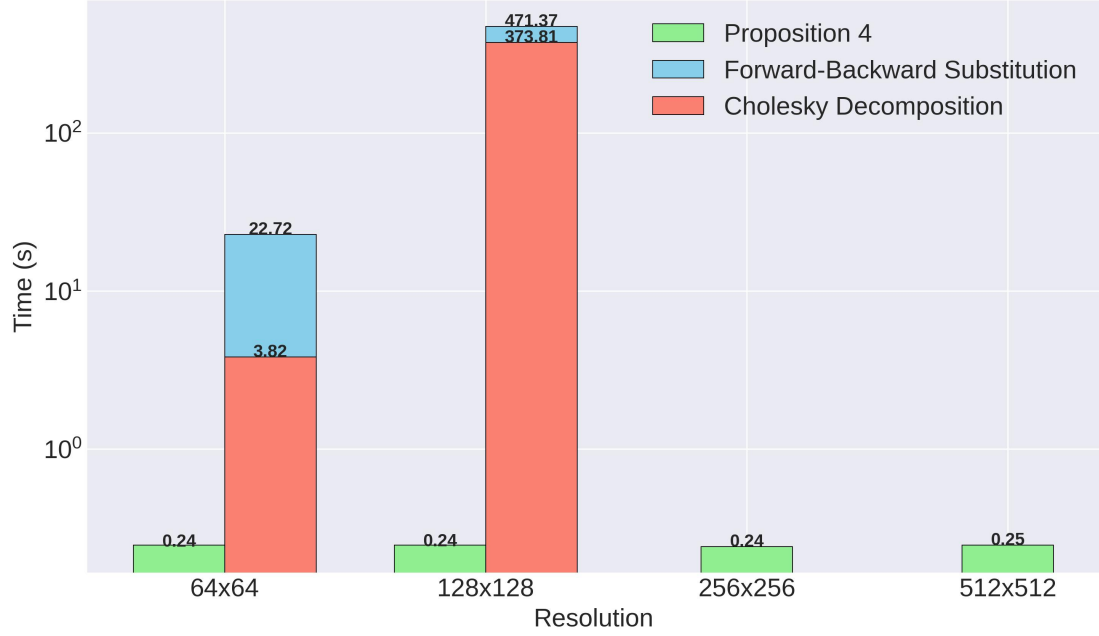


Figure 4.2: Comparison of solving the linear system in (4.94) using Proposition 4.7 and sparse Cholesky decomposition. The time for Cholesky-based solving is split into two parts: decomposition (orange) and forward-backward substitution (blue).

For resolutions 256×256 and 512×512 , Cholesky decomposition failed due to out-of-memory errors during testing.

Note that the worst-case computational complexity of reconstructing the transport plan via Algorithm 9 is $3m^2$. In practice, we can efficiently parallelize the (k, j) -loop to leverage the significant benefits of GPU acceleration, enabling an efficient reconstruction of the transport plan from a solution to the reduced OT model. We provide some additional numerical experiment results in Table 4.3 to better demonstrate the efficiency of the transport plan recovery with the GPU acceleration. The results shown in Table 4.3 demonstrate that our approach outperforms other popular algorithms, even when accounting for the plan recovery time. We have also considered the sparsity in the transport plan recovery, which facilitates handling large-scale problems.

Table 4.3: Numerical results of various algorithms on the DOTmark dataset: total time = solve time + transport plan recovery time.

Category	Resolution		HOT	Network Simplex	Gurobi	ADMM	Convolutional Sinkhorn	Sinkhorn
Classic	64 × 64	time(s)	0.67 (+0.17)	2.73	2.16 (+0.17)	1.77 (+0.17)	16.18	174.82
	128 × 128	time(s)	1.58 (+0.67)	36.18	29.15 (+0.67)	3.53 (+0.67)	39.40	2632.17
Shapes	64 × 64	time(s)	0.64 (+0.17)	1.48	1.33 (+0.17)	3.92 (+0.17)	9.60	103.74
	128 × 128	time(s)	1.68 (+0.67)	20.70	22.46 (+0.67)	2.32 (+0.67)	24.32	1616.34

The numerical comparison of the HOT algorithm on GPU and CPU is presented in Table 4.4. The results demonstrate that the GPU’s acceleration is substantial, and the acceleration effect becomes more significant as the problem’s dimension increases.

Table 4.4: Comparison of HOT’s performance on CPU and GPU.

Category	Resolution	Metric	GPU	CPU	Ratio ($t_{\text{CPU}}/t_{\text{GPU}}$)
Classic	128 × 128	time (s)	1.58	5.91	3.74
	256 × 256	time (s)	12.98	108.98	8.40
	512 × 512	time (s)	81.02	764.22	9.43
Shapes	128 × 128	time (s)	1.68	6.15	3.66
	256 × 256	time (s)	14.87	130.81	8.80
	512 × 512	time (s)	87.12	812.82	9.33

To better demonstrate the significance of the Halpern acceleration, we conduct additional numerical testing by changing the stopping criterion of the algorithm from relative KKT residual to absolute KKT residual and setting the maximum number of iterations to 1E6. The results are shown in Table 4.5 below.

Table 4.5: Numerical results of HOT and ADMM using absolute KKT residual as the stopping criterion (10^{-6}).

Category	Resolution	Metric	HOT	ADMM
Classic	64×64	time (s)	6.39	406.94
		gap	6.93E-10	1.58E-6
		feaserr	3.12E-13	4.51E-10
		iter	17680	1000000
	128×128	time (s)	76.36	1093.10
		gap	8.10E-10	4.69E-6
		feaserr	2.82E-14	6.11E-10
		iter	57300	1000000
	256×256	time (s)	3774.04	9135.66
		feaserr	3.96E-15	9.31E-11
		iter	338240	1000000
Shapes	64×64	time (s)	6.34	404.29
		gap	2.29E-10	6.48E-7
		feaserr	3.62E-13	8.78E-10
		iter	17660	991730
	128×128	time (s)	54.86	1087.06
		gap	2.47E-9	2.43E-6
		feaserr	2.33E-14	1.01E-9
		iter	41190	1000000
	256×256	time (s)	2519.99	9101.29
		feaserr	3.24E-15	1.10E-10
		iter	226020	1000000

4.4.2 An Application in Color Transfer

Color transfer (Pitié and Kokaram, 2007; Bonneel et al., 2013; Solomon et al., 2015) based on the OT model has important applications in various fields, including digital image processing, computer graphics, and visual arts. It involves transferring the color characteristics from a target image to a source image to achieve a desired visual effect. Inspired by its convincing performance in color grading and color histogram manipulation (Bonneel et al., 2013; Solomon et al., 2015), we conduct the color transfer over the CIE-Lab domain by applying the OT model to the 1D luminance channel and the 2D chrominance channel independently. Note that the 1D OT plan can be found efficiently (Villani, 2003). To efficiently conduct the OT over the 2D chrominance channel, we first solve the equivalent reduced OT model using the HOT Algorithm 10 and then recover an OT plan using Algorithm 9. To construct the supports of the two-dimensional histograms used in the reduced OT model, we apply the K-means algorithm with $K = 128$ to all chrominance values in the CIE-Lab color space that appear in the source image or the target image. For each resulting centroid, we draw a vertical and a horizontal line. The intersections of these lines form a set of (non-uniform) bins, which serve as the histogram supports. The performance of color transfer for selected image pairs can be found in Figure 4.3.

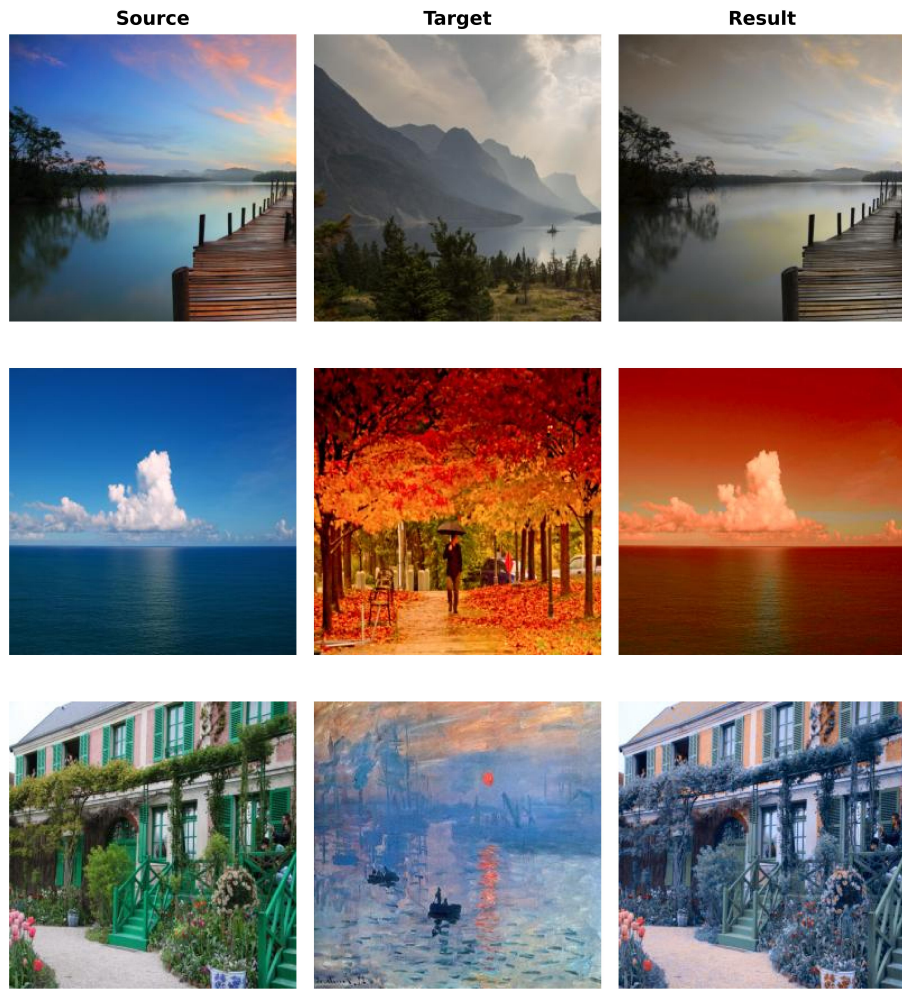


Figure 4.3: Selected examples of color transfer based on the reduced OT model with the OT plan recovered by Algorithm 9.

Chapter 5

Extension–HPR-LP: An Implementation of an HPR Method for Solving Linear Programming

This chapter presents HPR-LP, a practical implementation of the HPR method tailored for solving large-scale general LPs. HPR-LP incorporates adaptive restart strategies and penalty parameter updates. We evaluate its performance on benchmark LP datasets, demonstrating improvements in speed and robustness compared to leading solvers such as PDLP.

5.1 An HPR Method for Solving General LP Problems

Consider the following general LP problems:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \langle c, x \rangle \\ \text{s.t.} \quad & A_1 x = b_1, \\ & A_2 x \geq b_2, \\ & x \in C, \end{aligned} \tag{5.1}$$

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 chapter, we assume that A is a non-zero matrix. Then, the dual of problem (5.1) is given by

$$\begin{aligned} \min_{y \in \mathbb{R}^m, z \in \mathbb{R}^n} \quad & -\langle b, y \rangle + \delta_D(y) + \delta_C^*(-z) \\ \text{s.t.} \quad & A^*y + z = c, \end{aligned} \tag{5.2}$$

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 (Gurobi Optimization, LLC, 2024; IBM, 1987) 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 (Applegate et al., 2021, 2023; Basu et al., 2020; Deng et al., 2025; Lin et al., 2021; Lu and Yang, 2024; O'Donoghue, 2021; O'Donoghue et al., 2016; Stellato et al., 2020). Notably, Applegate et al. (Applegate et al., 2021, 2023) developed an award-winning solver PDLP¹, which employed the primal-dual hybrid gradient (PDHG) method (Zhu and Chan, 2008) as its base algorithm, equipped with several effective implementation techniques, including the use of the ergodic iterate as a restart point², an update rule for the penalty parameter σ , and a line search strategy. The GPU implementation of PDLP (cuPDLP.jl (Lu and Yang, 2025) and cuPDLP-c (Lu et al., 2023)) has shown some advantages over commercial LP solvers like Gurobi (Gurobi Optimization, LLC, 2024) and COPT (Ge et al., 2022) for solving large-scale LP

¹ The authors of (Applegate et al., 2021, 2023) 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., (Lu and Yang, 2025) and (Lu et al., 2023)), 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 (Applegate et al., 2023).

problems. It is worth noting that the ergodic sequence of the semi-proximal ADMM (sPADMM) (Fazel et al., 2013), including PDHG (Chambolle and Pock, 2011; Esser et al., 2010), achieves an $O(1/k)$ iteration complexity with respect to the objective error and the feasibility violation (Cui et al., 2016), 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 Compared to existing algorithms for solving LP, the HPR method introduced in Chapter 4 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.

According to (Rockafellar, 1970, Corollary 28.3.1), we know that $(y^*, z^*) \in \mathbb{R}^m \times \mathbb{R}^n$ is an optimal solution to problem (5.2) if there exists $x^* \in \mathbb{R}^n$ such that (y^*, z^*, x^*) satisfies the following KKT system:

$$0 \in Ax^* - b + \mathcal{N}_D(y^*), \quad 0 \in z^* + \mathcal{N}_C(x^*), \quad A^*y^* + z^* - c = 0. \quad (5.3)$$

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 (5.2) 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 Chapter 4 for solving problems (5.1) and (5.2) is presented in Algorithm 11.

³ Shen and Pan (Shen and Pan, 2016) extended Monteiro and Svaiter's ergodic $O(1/k)$ iteration complexity result with respect to the ε -subdifferential residual (Monteiro and Svaiter, 2013) from ADMM to sPADMM, which can yield an $O(1/\sqrt{k})$ iteration complexity for the KKT residual of LP problems.

Algorithm 11 An HPR method with semi-proximal terms for the problem (5.2)

- 1: **Input:** Set the penalty parameter $\sigma > 0$. Let $\mathcal{T}_1 : \mathbb{R}^m \rightarrow \mathbb{R}^m$ 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** $k = 0, 1, \dots$, **do**
 - 3: Step 1. $\bar{z}^{k+1} = \arg \min_{z \in \mathbb{R}^n} \{L_\sigma(y^k, z; x^k)\}$
 - 4: Step 2. $\bar{x}^{k+1} = x^k + \sigma(A^*y^k + \bar{z}^{k+1} - c)$
 - 5: Step 3. $\bar{y}^{k+1} = \arg \min_{y \in \mathbb{R}^m} \left\{L_\sigma(y, \bar{z}^{k+1}; \bar{x}^{k+1}) + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_1}^2\right\}$
 - 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: **Output:** Iteration sequence $\{\bar{w}^k\}$.
-

Remark 5.1. *Algorithm 11 corresponds to the accelerated preconditioned (semi-proximal) ADMM (pADMM) introduced in Algorithm 8 with $\alpha = 2$, where Step 5 is the Halpern iteration with a stepsize of $\frac{1}{k+2}$ (Halpern, 1967; Lieder, 2021). We refer to it as an HPR method with semi-proximal terms because, for the COP (2.12), the Halpern accelerating pADMM ($\alpha = 2$) without proximal terms in Algorithm 8 is equivalent to the HPR method without proximal terms in Algorithm 4 found in Proposition 4.3.*

Remark 5.2. *More recently, Lu et al. (2025) adopted the Halpern iteration to enhance the performance of PDLP (Applegate et al., 2021) for solving the LP problem (5.1), leading to the development of cuPDLPx. The core routine of cuPDLPx corresponds to a special case of Algorithm 11. Specifically, starting from an initial point $u^0 = (y^0, x^0) \in \mathbb{R}^m \times \mathbb{R}^n$, the routine of cuPDLPx with reflection parameter $\gamma \in [0, 1]$ is given by:*

$$\begin{cases} \text{PDHG}(u^k) = \begin{cases} \bar{x}^{k+1} = \Pi_C \left(x^k - \frac{\eta}{\omega} (c - A^* y^k) \right), \\ \bar{y}^{k+1} = \Pi_D \left(y^k + \eta \omega (b - A(2\bar{x}^{k+1} - x^k)) \right), \end{cases} \\ u^{k+1} = \frac{k+1}{k+2} \left((1 + \gamma) \text{PDHG}(u^k) - \gamma u^k \right) + \frac{1}{k+2} u^0. \end{cases} \quad (5.4)$$

Here, $\eta = \frac{0.99}{\|A\|_2}$ is the stepsize, and ω is a scalar parameter referred to as the primal weight. On the other hand, taking $\mathcal{T}_1 = \lambda_A I_m - AA^\top$ with $\lambda_A \geq \lambda_1(AA^*)$, Algorithm 11 simplifies to:

$$\begin{cases} \bar{z}^{k+1} = \frac{1}{\sigma} \left(\Pi_C \left(x^k + \sigma(A^* y^k - c) \right) - (x^k + \sigma(A^* y^k - c)) \right), \\ \bar{x}^{k+1} = x^k + \sigma(A^* y^k + \bar{z}^{k+1} - c) = \Pi_C \left(x^k + \sigma(A^* y^k - c) \right), \\ \bar{y}^{k+1} = \Pi_D \left(y^k + \frac{1}{\sigma \lambda_A} (b - A(2\bar{x}^{k+1} - x^k)) \right), \\ \hat{w}^{k+1} = 2\bar{w}^{k+1} - w^k, \\ w^{k+1} = \frac{1}{k+2} w^0 + \frac{k+1}{k+2} \hat{w}^{k+1}. \end{cases} \quad (5.5)$$

Comparing the update schemes (5.4) with $\gamma = 1$ and (5.5), we observe that the two methods produce identical iterates when the parameters satisfy the relations

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

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

$$\sigma = \frac{\eta}{\omega}, \quad \lambda_A = \frac{1}{\eta^2} \geq \|A\|_2^2, \quad (5.6)$$

then the sequence $\{u^k = (y^k, x^k)\}$ generated by (5.4) with $\gamma = 1$ coincides with the sequence $\{(y^k, x^k)\}$ produced by (5.5). In other words, the base iteration of cuPDLPx in (5.4) can be viewed as a specific instance of the Algorithm 11, under an appropriate choice of parameters and initialization.

Now, to discuss the global convergence of the HPR method with semi-proximal terms for problems (5.1) and (5.2) presented in Algorithm 11, we make the following assumption:

Assumption 5. *There exists a vector $(y^*, z^*, x^*) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ satisfying the KKT system (5.3).*

Under Assumption 5, solving problems (5.1) and (5.2) 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

$$\mathcal{T}w = \begin{pmatrix} -b + \mathcal{N}_D(y) + Ax \\ -\partial\delta_C^*(-z) + x \\ c - A^*y - z \end{pmatrix} \quad \forall w = (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n. \quad (5.7)$$

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

$$\mathcal{M} = \begin{bmatrix} \sigma AA^* + \sigma \mathcal{T}_1 & 0 & A \\ 0 & 0 & 0 \\ A^* & 0 & \frac{1}{\sigma} I_n \end{bmatrix}, \quad (5.8)$$

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 in the following proposition.

Proposition 5.1. *Consider the operators $\mathcal{T}$ defined in (5.7) and $\mathcal{M}$ defined in (5.8). Then the sequence $\{w^k\}$ generated by the HPR method with semi-proximal terms in Algorithm 11 coincides with the sequence $\{w^k\}$ generated by the following accelerated dPPM*

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

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 Proposition 4.2. $\square$

By utilizing the equivalence in Proposition 5.1, from Corollary 4.3 we can establish the global convergence of the HPR method with semi-proximal terms, as presented in the following proposition.

Proposition 5.2. *Suppose that Assumption 5 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 11 converges to the point $w^* = (y^*, z^*, x^*)$, where (y^*, z^*) solves problem (5.2) and x^* solves problem (5.1).*

Moreover, from Proposition 4.1, we have the following iteration complexity for the HPR method with semi-proximal terms:

Proposition 5.3. *Suppose that Assumption 5 holds. Then the sequences $\{w^k\}$ and $\{\hat{w}^k\}$ generated by the HPR method with semi-proximal terms in Algorithm 11 satisfy*

$$\|w^k - \hat{w}^{k+1}\|_{\mathcal{M}} \leq \frac{2\|w^0 - w^*\|_{\mathcal{M}}}{k+1} \quad \forall k \geq 0 \text{ and } w^* \in \mathcal{T}^{-1}(0). \quad (5.9)$$

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 (5.3), as introduced in (Han et al., 2018):

$$\mathcal{R}(w) = \begin{pmatrix} y - \Pi_D(y - Ax + b) \\ x - \Pi_C(x - z) \\ c - A^*y - z \end{pmatrix} \quad \forall w = (y, z, x) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n. \quad (5.10)$$

Additionally, let $\{(\bar{y}^k, \bar{z}^k)\}$ be the sequence generated by Algorithm 11. We define the objective error as follows:

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

where (y^*, z^*) is the limit point of the sequence $\{(\bar{y}^k, \bar{z}^k)\}$. Based on the iteration complexity presented in Proposition 5.3, 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 5.1. *Suppose that Assumption 5 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 11, 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:*

$$\|\mathcal{R}(\bar{w}^{k+1})\| \leq \left(\frac{\sigma(\|A\| + \|\sqrt{\mathcal{T}_1}\|) + 1}{\sqrt{\sigma}} \right) \frac{R_0}{(k+1)} \quad (5.11)$$

and

$$\left(\frac{-1}{\sqrt{\sigma}} \|x^*\| \right) \frac{R_0}{k+1} \leq h(\bar{y}^{k+1}, \bar{z}^{k+1}) \leq \left(3R_0 + \frac{1}{\sqrt{\sigma}} \|x^*\| \right) \frac{R_0}{k+1}. \quad (5.12)$$

Remark 5.3. *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 (2013) established an ergodic $O(1/k)$ iteration complexity for ADMM with a unit dual stepsize in terms of the ε -subdifferential, which 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 (Chambolle and Pock, 2011, 2016), Applegate et al. (Applegate et al., 2023) 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 non-ergodic sense, Cui et al. (2016) showed that a majorized ADMM with semi-proximal terms achieves a non-ergodic $O(1/\sqrt{k})$ 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 (5.11), providing stronger theoretical guaran-*

tees that, in turn, support practical advantages in solving large-scale LP problems efficiently.

5.2 An HPR Solver for LP Problems

In this section, we present an HPR-LP solver for solving large-scale LP problems, as outlined in Algorithm 12. 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.

Algorithm 12 HPR-LP: An HPR method for the LP problem (5.2)

- 1: **Input:** Let $\mathcal{T}_1 : \mathbb{R}^m \rightarrow \mathbb{R}^m$ 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 $w^{0,0} = (y^{0,0}, z^{0,0}, x^{0,0}) \in D \times \mathbb{R}^n \times \mathbb{R}^n$.
 - 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: $\bar{z}^{r,t+1} = \arg \min_{z \in \mathbb{R}^n} \{L_{\sigma_r}(y^{r,t}, z; x^{r,t})\}$
 - 7: $\bar{x}^{r,t+1} = x^{r,t} + \sigma_r(A^*y^{r,t} + \bar{z}^{r,t+1} - c)$
 - 8: $\bar{y}^{r,t+1} = \arg \min_{y \in \mathbb{R}^m} \left\{ L_{\sigma_r}(y, \bar{z}^{r,t+1}; \bar{x}^{r,t+1}) + \frac{\sigma_r}{2} \|y - y^{r,t}\|_{\mathcal{T}_1}^2 \right\}$
 - 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: $\sigma_{r+1} = \text{SigmaUpdate}(\bar{w}^{r,\tau_r}, w^{r,0}, \mathcal{T}_1, A), r = r + 1$
 - 15: **until** termination criteria hold
 - 16: **Output:** $\{\bar{w}^{r,t}\}$.
-

5.2.1 Restart Criteria

From Theorem 5.1, 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 complexity analysis in Proposition 5.3, 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 (5.3). Note that $R_{r,0}$ represents the upper bound (disregarding the factor 2) in the complexity results in Proposition 5.3 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 (Applegate et al., 2021; Lu and Yang, 2025; Lu et al., 2023).⁴ In addition, since w^* is unknown, we approximate $R_{r,t}$ using the following expression, inspired by Proposition 5.3:

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

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

1. Sufficient decay of $\tilde{R}_{r,t+1}$:

$$\tilde{R}_{r,t+1} \leq \alpha_1 \tilde{R}_{r,0}; \tag{5.13}$$

⁴ In (Applegate et al., 2021), 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 (Lu and Yang, 2025; Lu et al., 2023).

2. Necessary decay + no local progress of $\tilde{R}_{r,t+1}$:

$$\tilde{R}_{r,t+1} \leq \alpha_2 \tilde{R}_{r,0} \quad \text{and} \quad \tilde{R}_{r,t+1} > \tilde{R}_{r,t}; \quad (5.14)$$

3. Long inner loop:

$$t \geq \alpha_3 k, \quad (5.15)$$

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} .

5.2.2 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 5.3. Specifically, we update σ_{r+1} at the $(r + 1)$ -th restart for any $r \geq 0$ by solving the following optimization problem:

$$\sigma_{r+1} := \arg \min_{\sigma} \|w^{r+1,0} - w^*\|_{\mathcal{M}}^2, \quad (5.16)$$

where $\|w^{r+1,0} - w^*\|_{\mathcal{M}}$ represents the upper bound of the complexity results in Proposition 5.3 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 $t \geq 0$, which further results in a smaller KKT residual $\|\mathcal{R}(\bar{w}^{r+1,t+1})\|$. Substituting the definition of $\mathcal{M}$ from (5.8) into (5.16), we derive

$$\begin{aligned} \sigma_{r+1} &= \arg \min_{\sigma} \|w^{r+1,0} - w^*\|_{\mathcal{M}}^2 \\ &= \arg \min_{\sigma} (\sigma \|y^{r+1,0} - y^*\|_{\mathcal{T}_1}^2 + \sigma^{-1} \|x^{r+1,0} - x^* + \sigma A^*(y^{r+1,0} - y^*)\|^2) \\ &= \sqrt{\frac{\|x^{r+1,0} - x^*\|^2}{\|y^{r+1,0} - y^*\|_{\mathcal{T}_1}^2 + \|A^*(y^{r+1,0} - y^*)\|^2}}. \end{aligned} \quad (5.17)$$

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

$$\Delta_x := \|\bar{x}^{r,\tau_r} - x^{r,0}\| \quad \text{and} \quad \Delta_y := \sqrt{\|\bar{y}^{r,\tau_r} - y^{r,0}\|_{\mathcal{T}_1}^2 + \|A^*(\bar{y}^{r,\tau_r} - y^{r,0})\|^2}, \quad (5.18)$$

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

$$\sigma_{r+1} = \frac{\Delta_x}{\Delta_y}. \quad (5.19)$$

Note that the approximations Δ_x and Δ_y may deviate significantly from the true values, so we update σ using formula (5.19) only when the following conditions are met; otherwise, we reset σ to 1:

1. Δ_x and Δ_y are within a suitable range:

$$\Delta_x \in (10^{-16}, 10^{12}), \quad \Delta_y \in (10^{-16}, 10^{12}); \quad (5.20)$$

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

$$\frac{\text{error}_d}{\text{error}_p} \in (10^{-8}, 10^8), \quad (5.21)$$

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 13.

Algorithm 13 SigmaUpdate

- 1: **Input:** $(\bar{w}^{r,\tau_r}, w^{r,0}, \mathcal{T}_1, A)$
 - 2: Calculate Δ_x and Δ_y defined in (5.18)
 - 3: **if** conditions (5.20) and (5.21) are satisfied **then**
 - 4: $\sigma_{r+1} = \frac{\Delta_x}{\Delta_y}$
 - 5: **else**
 - 6: $\sigma_{r+1} = 1$
 - 7: **Output:** σ_{r+1}
-

Remark 5.4. *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 for solving more general COPs (2.12).*

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 (5.19) to determine σ .

1. $\mathcal{T}_1 = 0$. For example, consider the LP problems without inequality constraints (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:

$$AA^*\bar{y}^{r,t+1} = \frac{1}{\sigma_r}(b - A(\bar{x}^{r,t+1} + \sigma_r(\bar{z}^{r,t+1} - c))). \quad (5.22)$$

Solving the linear equations (5.22) by the direct method is affordable in many applications such as in the OT problem or the Wasserstein barycenter problem. According to (5.19), if conditions (5.20) and (5.21) are satisfied, then σ_{r+1} is computed as:

$$\sigma_{r+1} = \frac{\|\bar{x}^{r,\tau_r} - x^{r,0}\|}{\|A^*(\bar{y}^{r,\tau_r} - y^{r,0})\|}. \quad (5.23)$$

2. $\mathcal{T}_1 = \lambda_A I_m - AA^*$ with $\lambda_A \geq \lambda_1(AA^*)$ as proposed in (Esser et al., 2010; Chambolle and Pock, 2011; Xu and Wu, 2011). This applies when inequality constraints are present (i.e., $m_2 > 0$) or when solving the linear equations (5.22) directly is not affordable. In this case,

$$\begin{cases} \bar{y}_1^{r,t+1} = y_1^{r,t} + \frac{1}{\lambda_A}(b_1/\sigma_r - A_1 R_y), \\ \bar{y}_2^{r,t+1} = \Pi_{\mathbb{R}_+^{m_2}} \left(y_2^{r,t} + \frac{1}{\lambda_A}(b_2/\sigma_r - A_2 R_y) \right), \end{cases} \quad (5.24)$$

where $R_y := \bar{x}^{r,t+1}/\sigma_r + (A^*y^{r,t} + \bar{z}^{r,t+1} - c)$. If conditions (5.20) and (5.21) are satisfied, then σ_{r+1} is given by:

$$\sigma_{r+1} = \frac{1}{\sqrt{\lambda_A}} \frac{\|\bar{x}^{r,\tau_r} - x^{r,0}\|}{\|\bar{y}^{r,\tau_r} - y^{r,0}\|}. \quad (5.25)$$

Remark 5.5. The formula (5.25) is similar to the primal weight ω update formula with $\theta = 1$ in Algorithm 3 of (Applegate et al., 2021), except for the inclusion of the term λ_A .

5.2.3 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:

$$\begin{aligned}\bar{z}^{r,t+1} &= \arg \min_{z \in \mathbb{R}^n} \{L_{\sigma_r}(y^{r,t}, z; x^{r,t})\} \\ &= \frac{1}{\sigma_r} \left(\Pi_C(x^{r,t} + \sigma_r(A^*y^{r,t} - c)) - (x^{r,t} + \sigma_r(A^*y^{r,t} - c)) \right).\end{aligned}\quad (5.26)$$

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

$$\bar{x}^{r,t+1} = x^{r,t} + \sigma_r(A^*y^{r,t} + \bar{z}^{r,t+1} - c) = \Pi_C(x^{r,t} + \sigma_r(A^*y^{r,t} - c)). \quad (5.27)$$

For general LP problems, we set $\mathcal{T}_1 = \lambda_A I_m - AA^*$ with $\lambda_A \geq \lambda_1(AA^*)$ in the HPR-LP. Consequently, following from (5.24) and (5.27), the update for $\bar{y}^{r,t+1}$ is given by:

$$\begin{cases} \bar{y}_1^{r,t+1} = y_1^{r,t} + \frac{1}{\lambda_A \sigma_r} \left(b_1 - A_1(2\bar{x}^{r,t+1} - x^{r,t}) \right), \\ \bar{y}_2^{r,t+1} = \Pi_{\mathbb{R}_+^{m_2}} \left(y_2^{r,t} + \frac{1}{\lambda_A \sigma_r} \left(b_2 - A_2(2\bar{x}^{r,t+1} - x^{r,t}) \right) \right).\end{cases}\quad (5.28)$$

By combining (5.26), (5.27), and (5.28), 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 (5.26) when checking the termination criteria for stopping HPR-LP. Furthermore, the update of (5.26), (5.27), and (5.28) 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 (5.26), (5.27), and (5.28). For matrix-vector multiplications, we utilize `cusparseSpMV()` from the cuSPARSE library, which applies the `CUSPARSE_SPMV_CSR_ALG2` algorithm to ensure deterministic results.

5.3 Numerical Experiments

In this section, we compare the performance of HPR-LP implemented in Julia (Bezanson et al., 2017) with cuPDLP.jl (Lu and Yang, 2025) on a GPU. The experimental setup is detailed in Subsection 5.3.1. Subsection 5.3.2 discusses the performance of these two solvers on Mittelman’s LP benchmark set.⁵ Subsection 5.3.3 presents numerical results on LP relaxations of instances from the MIPLIB 2017 collection (Gleixner et al., 2021). Finally, Subsection 5.3.4 highlights the numerical results on extremely large instances, including LPs generated from quadratic assignment problems (QAPs) (Burkard et al., 1997), the “zib03” instance as discussed by Koch et al. (Koch et al., 2022), and the LP formulation of the PageRank problem (Nesterov, 2014).

5.3.1 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 (Lu and Yang, 2025) on two classic benchmark datasets: Mittelman’s LP benchmark set and LP relaxations of instances from the MIPLIB 2017 collection. We use 49 publicly available Mittelman’s LP benchmark instances. For the MIPLIB 2017 set, we initially select 383 instances based on the criteria outlined in (Lu and Yang, 2025). After excluding three instances during Gurobi’s

⁵ <https://plato.asu.edu/ftp/lpfeas.html>.

presolve (Gurobi Optimization, LLC, 2024), we retain 380 instances.⁶ We then split the MIP relaxations into three classes based on the number of nonzeros in the constraint matrix, as shown in Table 5.1, following the approach in (Lu and Yang, 2025). In addition, to further explore the limits of the capability of HPR-LP, we employ several extremely large-scale LP problems referenced in (Lu et al., 2023), including LPs generated from QAPs (Burkard et al., 1997), the “zib03” instance from (Koch et al., 2022), and several instances of PageRank problem (Nesterov, 2014).

Table 5.1: 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 (Bezanson et al., 2017), referred to as HPR-LP.jl. Similarly, cuPDLP.jl (Lu and Yang, 2025), the GPU version of PDLP, is also implemented in Julia.⁷ All tested solvers are run on an NVIDIA 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 (Ruiz, 2001),

⁶ Two instances are identified as unbounded by Gurobi’s presolve, and one instance is solved by Gurobi’s presolve.

⁷ 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.

followed by the bidiagonal preconditioning used by Pock and Chambolle (Pock and Chambolle, 2011) with $\alpha = 1$. Finally, we normalize b and c divided by $\|b\| + 1$ and $\|c\| + 1$, respectively. For cuPDL.jl (Lu and Yang, 2025), 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 (Golub and Van Loan, 2013). In HPR-LP, the restart criteria are based on conditions (5.13), (5.14), and (5.15), with parameters $\alpha_1 = 0.2$, $\alpha_2 = 0.6$, and $\alpha_3 = 0.2$. The penalty parameter σ is updated using formula (5.25). For cuPDL.jl (Lu and Yang, 2025), 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 PDL (Applegate et al., 2021; Lu and Yang, 2025; Lu et al., 2023) are satisfied for the tolerance $\varepsilon \in (0, \infty)$:

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

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 cuPDL.jl uses its default setting of checking termination criteria every 64 iterations.

Time limit. In Subsection 5.3.2, we impose 15000 seconds as the time limit in Mittelman’s benchmark. For Subsection 5.3.3, 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 Subsection 5.3.4, 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 Mittelman’s benchmarks, to measure the performance of solvers on a collection of problems. Specifically, the shifted geometric mean is defined as

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

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 (Lu and Yang, 2025), 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.

5.3.2 Mittelman’s LP Benchmark Set

Mittelman’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 12). We then compare HPR-LP.jl with cuPDLP.jl on Mittelman’s LP benchmark set.

Figure 5.1 summarizes the relative impact of HPR-LP’s enhancements on 49 instances of Mittelman’s LP benchmark set without presolve at a tolerance of

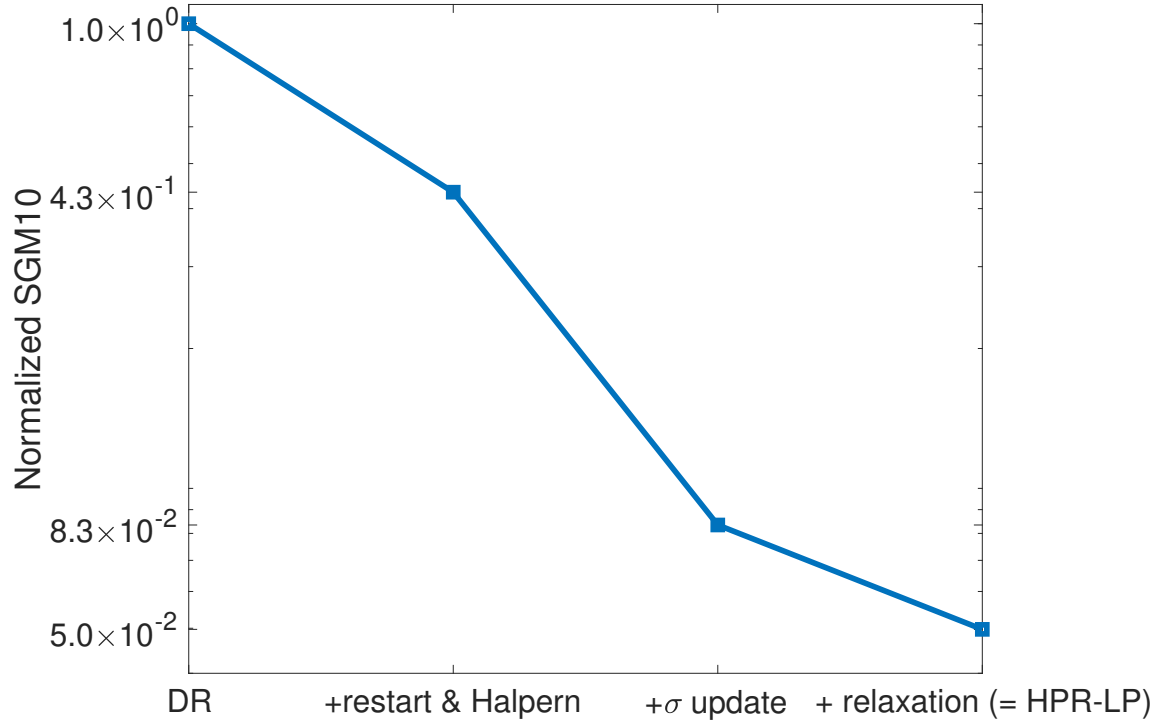
10^{-8} . As shown in subtable (a) of Figure 5.1, the enhancements described in Section 5.2 significantly improve the performance of the baseline Douglas-Rachford method, which corresponds to Steps 1–3 in Algorithm 11. Meanwhile, subfigure (b) of Figure 5.1 presents the normalized SGM10 values, computed 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** $\times$ 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** $\times$ speedup in terms of SGM10. Finally, by incorporating an additional relaxation step (Step 9 in Algorithm 12), HPR-LP benefits from a larger stepsize than HDR-LP, leading to a **1.67** $\times$ speedup over HDR-LP in terms of SGM10.

Figure 5.1: Relative impact of HPR-LP’s enhancements on Mittelmann’s LP benchmark set without presolve.

(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.



The numerical performance of HPR-LP.jl and cuPDLP.jl, both with and without presolve, is presented in Tables 5.2 and 5.3, 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 5.2 and 5.3.
- In terms of SGM10, HPR-LP.jl outperforms cuPDLP.jl on Mittelman’s LP benchmark set across all tolerance levels. For instance, as shown in Table 5.2, 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 5.3, 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 Mittelman’s LP benchmark set across all tolerance levels.

Table 5.2: Numerical performance of different solvers on 49 instances of Mittelman’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 5.3: Numerical performance of different solvers on 49 instances of Mittelman’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 Mittelman’s LP benchmark set in Table 5.4, similar to the SGM10 metric for time. As shown in Table 5.4, HPR-LP.jl requires fewer iterations than cuPDLP.jl across different tolerance levels. Additionally, Table 5.5 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 stepsize, which incurs significant computational overhead (Lu and Yang, 2025). In contrast, HPR-LP.jl makes use of the theoretical advance of the HPR method to choose the desirable relaxation parameter value of 2 (Step 9 in Algorithm (2)) to obtain a large stepsize, providing a guaranteed theoretical foundation for simpler and more efficient implementation.

Table 5.4: Shifted geometric mean of iteration counts for different solvers on 49 instances of Mittelman’s LP benchmark set, with and without presolve.

Solver	With Presolve			Without Presolve		
	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 5.5: Per-iteration time in seconds of different solvers for Mittelmann’s LP benchmark set.

Metric	With Presolve		Without Presolve	
	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

5.3.3 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 5.6 and 5.7, 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 5.6. Moreover, on the unpresolved dataset, HPR-LP.jl solves more problems than cuPDLP.jl does in different stopping tolerances, as shown in Table 5.7.
- In terms of SGM10 across 380 instances, HPR-LP.jl consistently outperforms cuPDLP.jl at all tolerance levels. As shown in Table 5.6, 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 5.7, HPR-LP.jl achieves a **2.03x** speedup over cuPDLP.jl to obtain a solution with the same accuracy for the unpresolved dataset.

Table 5.6: 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 5.7: 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

5.3.4 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 (Adams and Johnson, 1994) to generate LP instances of quadratic assignment problems (QAPs) from QAPLIB (Burkard et al., 1997), formulated as follows:

$$\begin{aligned}
& \min_{x,s} \quad \sum_{i,j} \sum_{k,l} a_{ik} b_{jl} s_{ijkl} \\
& \text{s.t.} \quad \sum_i s_{ijkl} = x_{kl}, \quad j, k, l = 1, \dots, N, \\
& \quad \sum_j s_{ijkl} = x_{kl}, \quad i, k, l = 1, \dots, N, \\
& \quad s_{ijkl} = s_{klij}, \quad i, j, k, l = 1, \dots, N, \\
& \quad s_{ijkl} \geq 0, \quad i, j, k, l = 1, \dots, N, \\
& \quad \sum_j x_{ij} = 1, \quad i = 1, \dots, N, \\
& \quad \sum_i x_{ij} = 1, \quad j = 1, \dots, N, \\
& \quad 0 \leq x_{ij} \leq 1, \quad i, j = 1, \dots, N,
\end{aligned} \tag{5.29}$$

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 5.8 and 5.9, 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 5.8. On the unpresolved dataset, as shown in Table 5.9, HPR-LP.jl achieves a **2.57x** speedup

over cuPDLP.jl in terms of SGM10 for the same relative accuracy.

Table 5.8: Solving time in seconds and SGM10 for different solvers on 20 QAP instances (Burkard et al., 1997) 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 5.9: Solving time in seconds and SGM10 for different solvers on 20 QAP instances (Burkard et al., 1997) 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 5.10 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 5.10: Solving time in seconds for the “zib03” instance (Koch et al., 2022).

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 (Lu et al., 2023).

Following the approach in (Applegate et al., 2021), 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 5.11. 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** $\times$ speedup over cuPDLP.jl when obtaining a solution with 10^{-8} relative accuracy, as shown in Table 5.11.

Table 5.11: 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

Chapter 6

Conclusions and Future Work

This thesis addressed the computational challenges of large-scale OT problems in the discrete setting by developing a suite of scalable, accelerated algorithms for solving linearly constrained COPs, which include many important discrete OT tasks—such as computing the KW distance and Wasserstein barycenters with pre-specified finite supports. Building on the strong performance of these algorithms for OT-related problems, we further extended the methodology to general large-scale LP problems.

We first proposed the HPR algorithm without proximal terms for solving COPs, derived by applying Halpern iteration to the Peaceman–Rachford splitting method. The algorithm achieved a non-ergodic iteration complexity of $O(1/\varepsilon)$ for computing an ε -approximate solution in terms of the KKT residual. When applied to the WBP with pre-specified finite supports involving T discrete probability distributions, each supported on m points, we derived closed-form solutions for the subproblems and attained an overall computational complexity of $O(Tm^2/\varepsilon)$ flops—improving upon the best-known bound. Extensive experiments on both synthetic and real datasets demonstrated the superior performance of the HPR algorithm for solving large-scale WBPs.

Second, we proposed an accelerated dPPA that achieved both asymptotic $o(1/k)$ and non-asymptotic $O(1/k)$ convergence rates by combining Halpern iteration with

the fast Krasnosel’skiĭ–Mann iteration. Exploiting the equivalence between pADMM and dPPA, we developed an accelerated pADMM algorithm with semi-proximal terms that achieves these convergence rates for both the KKT residual and the objective error. Based on this framework, we introduced HOT, an efficient Halpern accelerating algorithm for computing the KW distance via an equivalent reduced OT model. By designing a linear-time procedure to solve the involved linear systems, HOT achieved an ε -approximate solution in $O(m^{1.5}/\varepsilon)$ flops for discrete OT problems with m supports in $\mathbb{R}^2$, significantly improving upon the previous best-known bound. Furthermore, we designed a recovery scheme for obtaining the full transport plan from the reduced model, addressing a key limitation in applications requiring access to the transport map. Numerical experiments confirmed that HOT outperformed existing state-of-the-art algorithms in both accuracy and scalability.

Finally, motivated by the promising empirical performance of HPR for solving OT-related problems, we developed HPR-LP for solving large-scale general LPs. This solver incorporated adaptive restart strategies and dynamic penalty parameter updates into the HPR framework with semi-proximal terms. Extensive experiments on standard LP benchmark datasets showed that HPR-LP was efficient and robust across a wide range of solution tolerances, achieving significant speedups over the award-winning solver PDLP.

There are still many promising directions that could advance the development of accelerated algorithms and solver frameworks for OT and LP. Below, we briefly highlight several research avenues that merit further exploration.

- **Extension to nonconvex optimization.** Kong and Monteiro (2024) recently introduced a dampened proximal ADMM for solving linearly constrained, non-separable, nonconvex optimization problems. Their method achieves an approximate first-order stationary point within $O(\varepsilon^{-3})$ iterations under a basic

Slater condition. An exciting direction for future work is to explore whether the acceleration techniques developed in this thesis can be extended to design a novel accelerated pADMM for nonconvex problems, potentially improving upon the existing complexity bounds.

- **Generalization of HOT for OT and barycenter problems.** While the current HOT algorithm focuses on OT problems with discrete supports in $\mathbb{R}^2$, it would be valuable to generalize the implementation to higher-dimensional settings $\mathbb{R}^d$. Moreover, designing an efficient Halpern-accelerated algorithm for solving the Wasserstein barycenter problem with discrete supports remains an open and impactful research direction.
- **Enhancing the HPR-LP solver.** Although the HPR-LP solver shows strong empirical performance, there are still instances where it struggles to converge within the allocated time. One potential enhancement is to incorporate the fast Krasnosel’skiĭ–Mann iteration into the HPR framework, which may yield an improved iteration complexity of $o(1/k)$ in terms of the KKT residual and objective error. Another promising direction is to develop an algorithm with linear convergence better than that possessed by sPADMM (Han et al., 2018). Both directions offer promising opportunities for improving the theoretical and practical performance of large-scale LP solvers.
- **Using second-order methods to warm-start first-order methods.** For large-scale optimization problems such as LP problems, second-order methods typically rely on iterative solvers to handle Newton systems, which can become computationally expensive, particularly in the presence of problem degeneracy. A promising direction is to employ second-order methods in the early iterations to generate a high-quality warm start, and then switch to a more scalable

first-order method once solving the Newton systems becomes costly (e.g., due to increasing iteration counts of iterative solvers). This hybrid strategy may effectively combine the fast convergence of second-order methods with the low per-iteration cost and scalability of first-order algorithms.

Bibliography

- Adams, W. P. and Johnson, T. A. (1994), “Improved linear programming based lower bounds for the quadratic assignment problem,” in *Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993*, eds. P. M. Pardalos and H. Wolkowicz, American Mathematical Society.
- Agueh, M. and Carlier, G. (2011), “Barycenters in the Wasserstein space,” *SIAM J. Math. Anal.*, 43, 904–924.
- Applegate, D., Díaz, M., Hinder, O., Lu, H., Lubin, M., O’Donoghue, B., and Schudy, W. (2021), “Practical large-scale linear programming using primal-dual hybrid gradient,” *Advances in Neural Information Processing Systems*, 34, 20243–20257.
- Applegate, D., Hinder, O., Lu, H., and Lubin, M. (2023), “Faster first-order primal-dual methods for linear programming using restarts and sharpness,” *Math. Program.*, 201, 133–184.
- Auricchio, G., Bassetti, F., Gualandi, S., and Veneroni, M. (2018), “Computing Kantorovich-Wasserstein distances on d -dimensional histograms using $(d + 1)$ -partite graphs,” *Advances in Neural Information Processing Systems*, 31.
- Basu, K., Ghoting, A., Mazumder, R., and Pan, Y. (2020), “ECLIPSE: An extreme-scale linear program solver for web-applications,” in *International Conference on Machine Learning*, pp. 704–714, PMLR.
- Basu, S., Kolouri, S., and Rohde, G. K. (2014), “Detecting and visualizing cell phenotype differences from microscopy images using transport-based morphometry,” *Proc. Natl. Acad. Sci.*, 111, 3448–3453.
- Bauschke, H. H. and Combettes, P. L. (2017), *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd edn., Springer, New York.
- Beck, A. and Teboulle, M. (2009), “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM J. Imaging Sci.*, 2, 183–202.
- Benamou, J.-D., Carlier, G., Cuturi, M., Nenna, L., and Peyré, G. (2015), “Iterative Bregman projections for regularized transportation problems,” *SIAM J. Sci. Comput.*, 37, A1111–A1138.

- Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017), “Julia: A fresh approach to numerical computing,” *SIAM REV.*, 59, 65–98.
- Bigot, J. and Klein, T. (2018), “Characterization of barycenters in the Wasserstein space by averaging optimal transport maps,” *ESAIM: Probab. Stat.*, 22, 35–57.
- Blanchet, J., Jambulapati, A., Kent, C., and Sidford, A. (2024), “Towards optimal running times for optimal transport,” *Oper. Res. Lett.*, 52, 107054.
- Bonneel, N., Van De Panne, M., Paris, S., and Heidrich, W. (2011), “Displacement interpolation using Lagrangian mass transport,” in *Proceedings of the 2011 SIG-GRAPH Asia Conference*, pp. 1–12.
- Bonneel, N., Sunkavalli, K., Paris, S., and Pfister, H. (2013), “Example-based video color grading,” *ACM Trans. Graph.*, 32, 1–12.
- Bonneel, N., Peyré, G., and Cuturi, M. (2016), “Wasserstein barycentric coordinates: Histogram regression using optimal transport,” *ACM Trans. Graph.*, 35, 71:1–71:10.
- Boţ, R. I. and Nguyen, D.-K. (2023), “Fast Krasnosel’skiĭ-Mann algorithm with a convergence rate of the fixed point iteration of $o(\frac{1}{k})$,” *SIAM J. Numer. Anal.*, 61, 2813–2843.
- Boţ, R. I., Csetnek, E. R., and Nguyen, D.-K. (2025), “Fast optimistic gradient descent ascent (OGDA) method in continuous and discrete time,” *Found. Comput. Math.*, 25, 163–222.
- Bredies, K., Chenchene, E., Lorenz, D. A., and Naldi, E. (2022), “Degenerate preconditioned proximal point algorithms,” *SIAM J. Optim.*, 32, 2376–2401.
- Brézis, H. and Lions, P. L. (1978), “Produits infinis de résolvantes,” *Israel J. Math.*, 29, 329–345.
- Burkard, R. E., Karisch, S. E., and Rendl, F. (1997), “QAPLIB—a quadratic assignment problem library,” *J. Global Optim.*, 10, 391–403.
- Chambolle, A. and Contreras, J. P. (2022), “Accelerated Bregman primal-dual methods applied to optimal transport and Wasserstein barycenter problems,” *SIAM J. Math. Data Sci.*, 4, 1369–1395.
- Chambolle, A. and Pock, T. (2011), “A first-order primal-dual algorithm for convex problems with applications to imaging,” *J. Math. Imaging Vision*, 40, 120–145.
- Chambolle, A. and Pock, T. (2016), “On the ergodic convergence rates of a first-order primal-dual algorithm,” *Math. Program.*, 159, 253–287.

- Chen, C., Chan, R. H., Ma, S., and Yang, J. (2015), “Inertial proximal ADMM for linearly constrained separable convex optimization,” *SIAM J. Imaging Sci.*, 8, 2239–2267.
- Chen, L., Li, X., Sun, D. F., and Toh, K.-C. (2021), “On the equivalence of inexact proximal ALM and ADMM for a class of convex composite programming,” *Math. Program.*, 185, 111–161.
- Chiappori, P.-A., McCann, R. J., and Nesheim, L. P. (2010), “Hedonic price equilibria, stable matching, and optimal transport: Equivalence, topology, and uniqueness,” *Econ. Theory*, 42, 317–354.
- Claici, S., Chien, E., and Solomon, J. (2018), “Stochastic Wasserstein barycenters,” in *International Conference on Machine Learning*, pp. 999–1008, PMLR.
- Contreras, J. P. and Cominetti, R. (2023), “Optimal error bounds for non-expansive fixed-point iterations in normed spaces,” *Math. Program.*, 199, 343–374.
- Cotar, C., Friesecke, G., and Klüppelberg, C. (2013), “Density functional theory and optimal transportation with Coulomb cost,” *Commun. Pure Appl. Math.*, 66, 548–599.
- Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2016), “Optimal transport for domain adaptation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 39, 1853–1865.
- Cui, Y., Li, X., Sun, D. F., and Toh, K.-C. (2016), “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.
- Cuturi, M. (2013), “Sinkhorn distances: Lightspeed computation of optimal transport,” *Advances in Neural Information Processing Systems*, 26, 2292–2300.
- Cuturi, M. and Doucet, A. (2014), “Fast computation of Wasserstein barycenters,” in *International Conference on Machine Learning*, pp. 685–693, PMLR.
- Cuturi, M. and Peyré, G. (2016), “A smoothed dual approach for variational Wasserstein problems,” *SIAM J. Imaging Sci.*, 9, 320–343.
- Dantzig, G. B. and Thapa, M. N. (2003), *Linear Programming 2: Theory and Extensions*, Springer.
- Davis, D. and Yin, W. (2016), “Convergence rate analysis of several splitting schemes,” in *Splitting Methods in Communication, Imaging, Science, and Engineering*, pp. 115–163, Springer.

- Del Barrio, E., Cuesta-Albertos, J. A., Matrán, C., and Rodríguez-Rodríguez, J. M. (1999), “Tests of goodness of fit based on the L_2 -Wasserstein distance,” *Ann. Statist.*, pp. 1230–1239.
- Deng, Q., Feng, Q., Gao, W., Ge, D., Jiang, B., Jiang, Y., Liu, J., Liu, T., Xue, C., Ye, Y., and Zhang, C. (2025), “An enhanced alternating direction method of multipliers-based interior point method for linear and conic optimization,” *INFORMS J. Comput.*, 37, 338–359.
- Dominitz, A. and Tannenbaum, A. (2009), “Texture mapping via optimal mass transport,” *IEEE Trans. Vis. Comput. Graph.*, 16, 419–433.
- Dvinskikh, D. and Tiapkin, D. (2021), “Improved complexity bounds in Wasserstein barycenter problem,” in *International Conference on Artificial Intelligence and Statistics*, pp. 1738–1746, PMLR.
- Dvurechenskii, P., Dvinskikh, D., Gasnikov, A., Uribe, C., and Nedich, A. (2018), “Decentralize and randomize: Faster algorithm for Wasserstein barycenters,” *Advances in Neural Information Processing Systems*, 31.
- Dvurechensky, P., Gasnikov, A., and Kroshnin, A. (2018), “Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn’s algorithm,” in *International Conference on Machine Learning*, pp. 1367–1376, PMLR.
- Eckstein, J. (1994), “Some saddle-function splitting methods for convex programming,” *Optim. Methods Softw.*, 4, 75–83.
- Eckstein, J. and Bertsekas, D. P. (1992), “On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators,” *Math. Program.*, 55, 293–318.
- Esser, E., Zhang, X., and Chan, T. F. (2010), “A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science,” *SIAM J. Imaging Sci.*, 3, 1015–1046.
- Fazel, M., Pong, T. K., Sun, D. F., and Tseng, P. (2013), “Hankel matrix rank minimization with applications to system identification and realization,” *SIAM J. Matrix Anal. Appl.*, 34, 946–977.
- Ferradans, S., Papadakis, N., Peyré, G., and Aujol, J.-F. (2014), “Regularized discrete optimal transport,” *SIAM J. Imaging Sci.*, 7, 1853–1882.
- Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V.,

- Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. (2021), “POT: Python Optimal Transport,” *J. Mach. Learn. Res.*, 22, 1–8.
- Gabay, D. and Mercier, B. (1976), “A dual algorithm for the solution of nonlinear variational problems via finite element approximation,” *Comput. Math. Appl.*, 2, 17–40.
- Gabow, H. N. and Tarjan, R. E. (1991), “Faster scaling algorithms for general graph matching problems,” *J. ACM.*, 38, 815–853.
- Galichon, A. (2016), *Optimal Transport Methods in Economics*, Princeton University Press.
- Ge, D., Wang, H., Xiong, Z., and Ye, Y. (2019), “Interior-point methods strike back: Solving the Wasserstein barycenter problem,” *Advances in Neural Information Processing Systems*, 32.
- Ge, D., Huangfu, Q., Wang, Z., Wu, J., and Ye, Y. (2022), “Cardinal Optimizer (COPT) User Guide,” *arXiv preprint arXiv:2208.14314*.
- Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001), “From few to many: Illumination cone models for face recognition under variable lighting and pose,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 23, 643–660.
- Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al. (2021), “MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library,” *Math. Program. Comput.*, 13, 443–490.
- Glowinski, R. and Marroco, A. (1975), “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, 41–76.
- Goldberg, A. V., Tardos, É., and Tarjan, R. (1989), “Network flow algorithm,” Tech. rep., Cornell University Operations Research and Industrial Engineering.
- Goldstein, T., O’Donoghue, B., Setzer, S., and Baraniuk, R. (2014), “Fast alternating direction optimization methods,” *SIAM J. Imaging Sci.*, 7, 1588–1623.
- Golub, G. H. and Van Loan, C. F. (2013), *Matrix Computations*, JHU press.
- Gramfort, A., Peyré, G., and Cuturi, M. (2015), “Fast optimal transport averaging of neuroimaging data,” in *International Conference on Information Processing in Medical Imaging*, pp. 261–272, Springer.

- Guminov, S., Dvurechensky, P., Tupitsa, N., and Gasnikov, A. (2021), “On a combination of alternating minimization and Nesterov’s momentum,” in *International Conference on Machine Learning*, pp. 3886–3898, PMLR.
- Gurobi Optimization, LLC (2024), “Gurobi Optimizer Reference Manual,” .
- Haker, S., Zhu, L., Tannenbaum, A., and Angenent, S. (2004), “Optimal mass transport for registration and warping,” *Int. J. Comput. Vis.*, 60, 225–240.
- Halpern, B. (1967), “Fixed points of nonexpanding maps,” *Bull. Amer. Math. Soc.*, 73, 957–961.
- Han, D., Sun, D. F., and Zhang, L. (2018), “Linear rate convergence of the alternating direction method of multipliers for convex composite programming,” *Math. Oper. Res.*, 43, 622–637.
- IBM (1987), “IBM ILOG CPLEX Optimization Studio CPLEX User’s Manual,” .
- Indyk, P. and Thaper, N. (2003), “Fast color image retrieval via embeddings,” in *Workshop on Statistical and Computational Theories of Vision (at ICCV)*.
- Jambulapati, A., Sidford, A., and Tian, K. (2019), “A direct $\tilde{O}(1/\epsilon)$ iteration parallel algorithm for optimal transport,” *Advances in Neural Information Processing Systems*, 32.
- Kantorovich, L. (1942), “On the transfer of masses (in Russian),” *Doklady Akademii Nauk*, 37, 227–229.
- Khamis, A., Tsuchida, R., Tarek, M., Rolland, V., and Petersson, L. (2024), “Scalable optimal transport methods in machine learning: A contemporary survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–20.
- Kim, D. (2021), “Accelerated proximal point method for maximally monotone operators,” *Math. Program.*, 190, 57–87.
- Koch, T., Berthold, T., Pedersen, J., and Vanaret, C. (2022), “Progress in mathematical programming solvers from 2001 to 2020,” *EURO J. Comput. Optim.*, 10, 100031.
- Kong, W. and Monteiro, R. D. (2024), “Global complexity bound of a proximal ADMM for linearly constrained nonseparable nonconvex composite programming,” *SIAM J. Optim.*, 34, 201–224.
- Kovács, P. (2015), “Minimum-cost flow algorithms: An experimental evaluation,” *Optim. Methods Softw.*, 30, 94–127.

- Krawtschenko, R., Uribe, C. A., Gasnikov, A., and Dvurechensky, P. (2020), “Distributed optimization with quantization for computing Wasserstein barycenters,” *arXiv preprint arXiv:2010.14325*.
- Kroshnin, A., Tupitsa, N., Dvinskikh, D., Dvurechensky, P., Gasnikov, A., and Uribe, C. (2019), “On the complexity of approximating Wasserstein barycenters,” in *International Conference on Machine Learning*, pp. 3530–3540, PMLR.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998), “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, 86, 2278–2324.
- Leeb, W. and Coifman, R. (2016), “Hölder–Lipschitz norms and their duals on spaces with semigroups, with applications to earth mover’s distance,” *J. Fourier Anal. Appl.*, 22, 910–953.
- Li, H. and Lin, Z. (2019), “Accelerated alternating direction method of multipliers: An optimal $O(1/K)$ nonergodic analysis,” *J. Sci. Comput.*, 79, 671–699.
- Li, J. and Wang, J. Z. (2008), “Real-time computerized annotation of pictures,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 30, 985–1002.
- Li, X., Sun, D. F., and Toh, K.-C. (2016), “A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions,” *Math. Program.*, 155, 333–373.
- Li, X., Sun, D. F., and Toh, K.-C. (2018), “QSDPNAL: A two-phase augmented Lagrangian method for convex quadratic semidefinite programming,” *Math. Program. Comput.*, 10, 703–743.
- Li, X., Sun, D. F., and Toh, K.-C. (2019), “A block symmetric Gauss–Seidel decomposition theorem for convex composite quadratic programming and its applications,” *Math. Program.*, 175, 395–418.
- Li, X., Sun, D. F., and Toh, K.-C. (2020), “An asymptotically superlinearly convergent semismooth Newton augmented Lagrangian method for linear programming,” *SIAM J. Optim.*, 30, 2410–2440.
- Liang, L., Li, X., Sun, D. F., and Toh, K.-C. (2022), “QPPAL: A two-phase proximal augmented Lagrangian method for high-dimensional convex quadratic programming problems,” *ACM Trans. Math. Software*, 48, 1–27.
- Lieder, F. (2021), “On the convergence rate of the Halpern-iteration,” *Optim. Lett.*, 15, 405–418.
- Lin, T., Ho, N., and Jordan, M. (2019), “On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms,” in *International Conference on Machine Learning*, pp. 3982–3991, PMLR.

- Lin, T., Ho, N., Chen, X., Cuturi, M., and Jordan, M. (2020), “Fixed-support Wasserstein barycenters: Computational hardness and fast algorithm,” *Advances in Neural Information Processing Systems*, 33, 5368–5380.
- Lin, T., Ma, S., Ye, Y., and Zhang, S. (2021), “An ADMM-based interior-point method for large-scale linear programming,” *Optim. Methods Softw.*, 36, 389–424.
- Lin, T., Ho, N., and Jordan, M. I. (2022), “On the efficiency of entropic regularized algorithms for optimal transport,” *J. Mach. Learn. Res.*, 23, 1–42.
- Ling, H. and Okada, K. (2007), “An efficient earth mover’s distance algorithm for robust histogram comparison,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 29, 840–853.
- Lions, P.-L. and Mercier, B. (1979), “Splitting algorithms for the sum of two nonlinear operators,” *SIAM J. Numer. Anal.*, 16, 964–979.
- Lu, H. and Yang, J. (2024), “Restarted Halpern PDHG for Linear Programming,” *arXiv preprint arXiv:2407.16144*.
- Lu, H. and Yang, J. (2025), “cuPDL.jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia,” *Oper. Res.*, 73, 3440–3452.
- Lu, H., Yang, J., Hu, H., Huangfu, Q., Liu, J., Liu, T., Ye, Y., Zhang, C., and Ge, D. (2023), “cuPDL-C: A Strengthened Implementation of cuPDL for Linear Programming by C language,” *arXiv preprint arXiv:2312.14832*.
- Lu, H., Peng, Z., and Yang, J. (2025), “cuPDLpx: A further enhanced GPU-based first-order solver for linear programming,” *arXiv preprint arXiv:2507.14051*.
- Mai, V. V., Lindbäck, J., and Johansson, M. (2022), “A fast and accurate splitting method for optimal transport: Analysis and implementation,” *International Conference on Learning Representations*.
- Monge, G. (1781), “Mémoire sur la théorie des déblais et des remblais,” *Histoire de l’Académie Royale des Sciences*, pp. 666–704.
- Monteiro, R. D. and Sim, C.-K. (2018), “Complexity of the relaxed Peaceman–Rachford splitting method for the sum of two maximal strongly monotone operators,” *Comput. Optim. Appl.*, 70, 763–790.
- Monteiro, R. D. and Svaiter, B. F. (2013), “Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers,” *SIAM J. Optim.*, 23, 475–507.
- Nene, S. A., Nayar, S. K., and Murase, H. (1996), “Columbia Object Image Library (coil-100),” Tech. rep., CUCS-006-96.

- Nesterov, Y. (1983), “A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$,” *Dokl. Akad. Nauk. SSSR*, 269, 543.
- Nesterov, Y. (2014), “Subgradient methods for huge-scale optimization problems,” *Math. Program.*, 146, 275–297.
- O’Donoghue, B. (2021), “Operator splitting for a homogeneous embedding of the linear complementarity problem,” *SIAM J. Optim.*, 31, 1999–2023.
- Orlin, J. (1988), “A faster strongly polynomial minimum cost flow algorithm,” in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pp. 377–387.
- Ouyang, Y., Chen, Y., Lan, G., and Pasiliao Jr, E. (2015), “An accelerated linearized alternating direction method of multipliers,” *SIAM J. Imaging Sci.*, 8, 644–681.
- O’Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2016), “Conic optimization via operator splitting and homogeneous self-dual embedding,” *J. Optim. Theory Appl.*, 169, 1042–1068.
- Pele, O. and Werman, M. (2009), “Fast and robust earth mover’s distances,” in *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, IEEE.
- Peyré, G. and Cuturi, M. (2019), “Computational Optimal Transport: With Applications to Data Science,” *Foundations and Trends® in Machine Learning*, 11, 355–607.
- Pitié, F. and Kokaram, A. (2007), “The linear Monge–Kantorovitch linear colour mapping for example-based colour transfer,” *4th European Conference on Visual Media Production*.
- Pock, T. and Chambolle, A. (2011), “Diagonal preconditioning for first order primal-dual algorithms in convex optimization,” in *2011 International Conference on Computer Vision*, pp. 1762–1769, IEEE.
- Polyak, B. T. (1987), *Introduction to Optimization*, Optimization Software, New York.
- Rockafellar, R. T. (1970), *Convex Analysis*, Princeton University Press, Princeton, NJ.
- Rockafellar, R. T. (1976a), “Augmented Lagrangians and applications of the proximal point algorithm in convex programming,” *Math. Oper. Res.*, 1, 97–116.
- Rockafellar, R. T. (1976b), “Monotone operators and the proximal point algorithm,” *SIAM J. Control Optim.*, 14, 877–898.

- Rockafellar, R. T. and Wets, R. J.-B. (1998), *Variational Analysis*, Springer, New York.
- Rogozin, A., Beznosikov, A., Dvinskikh, D., Kovalev, D., Dvurechensky, P., and Gasnikov, A. (2021), “Decentralized distributed optimization for saddle point problems,” *arXiv preprint arXiv:2102.07758*.
- Ruiz, D. (2001), “A scaling algorithm to equilibrate both rows and columns norms in matrices,” Tech. rep., Rutherford Appleton Laboratory.
- Sabach, S. and Shtern, S. (2017), “A first order method for solving convex bilevel optimization problems,” *SIAM J. Optim.*, 27, 640–660.
- Sabach, S. and Teboulle, M. (2022), “Faster Lagrangian-based methods in convex optimization,” *SIAM J. Optim.*, 32, 204–227.
- Schmitzer, B. (2019), “Stabilized sparse scaling algorithms for entropy regularized transport problems,” *SIAM J. Sci. Comput.*, 41, A1443–A1481.
- Schrieber, J., Schuhmacher, D., and Gottschlich, C. (2016), “Dotmark—a benchmark for discrete optimal transport,” *IEEE Access*, 5, 271–282.
- Shen, L. and Pan, S. (2016), “Weighted iteration complexity of the sPADMM on the KKT residuals for convex composite optimization,” *arXiv preprint arXiv:1611.03167*.
- Shirdhonkar, S. and Jacobs, D. W. (2008), “Approximate earth mover’s distance in linear time,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE.
- Solomon, J. (2018), “Optimal transport on discrete domains,” *AMS Short Course on Discrete Differential Geometry*.
- Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. (2015), “Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains,” *ACM Trans. Graph.*, 34, 1–11.
- Staib, M., Claici, S., Solomon, J. M., and Jegelka, S. (2017), “Parallel streaming Wasserstein barycenters,” *Advances in Neural Information Processing Systems*, 30.
- Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020), “OSQP: An operator splitting solver for quadratic programs,” *Math. Program. Comput.*, 12, 637–672.
- Tang, T. and Toh, K.-C. (2024), “Self-adaptive ADMM for semi-strongly convex problems,” *Math. Program. Comput.*, 16, 113–150.

- Tiapkin, D., Gasnikov, A., and Dvurechensky, P. (2022), “Stochastic saddle-point optimization for the Wasserstein barycenter problem,” *Optim. Lett.*, 16, 2145–2175.
- Tran-Dinh, Q. (2024), “From Halpern’s fixed-point iterations to Nesterov’s accelerated interpretations for root-finding problems,” *Comput. Optim. Appl.*, 87, 181–218.
- Tran-Dinh, Q. and Luo, Y. (2021), “Halpern-type accelerated and splitting algorithms for monotone inclusions,” *arXiv preprint arXiv:2110.08150*.
- Tupitsa, N., Dvurechensky, P., Dvinskikh, D., and Gasnikov, A. (2022), “Numerical methods for large-scale optimal transport,” *arXiv preprint arXiv:2210.11368*.
- Uribe, C. A., Dvinskikh, D., Dvurechensky, P., Gasnikov, A., and Nedić, A. (2018), “Distributed computation of Wasserstein barycenters over networks,” in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6544–6549, IEEE.
- Villani, C. (2003), “Topics in Optimal Transportation,” *Graduate Studies in Mathematics*.
- Wang, W., Slepčev, D., Basu, S., Ozolek, J. A., and Rohde, G. K. (2013), “A linear optimal transportation framework for quantifying and visualizing variations in sets of images,” *Int. J. Comput. Vis.*, 101, 254–269.
- Wittmann, R. (1992), “Approximation of fixed points of nonexpansive mappings,” *Arch. Math.*, 58, 486–491.
- Xiao, Y., Chen, L., and Li, D. (2018), “A generalized alternating direction method of multipliers with semi-proximal terms for convex composite conic programming,” *Math. Program. Comput.*, 10, 533–555.
- Xu, M. and Wu, T. (2011), “A class of linearized proximal alternating direction methods,” *J. Optim. Theory Appl.*, 151, 321–337.
- Xu, Y. (2017), “Accelerated first-order primal-dual proximal methods for linearly constrained composite convex programming,” *SIAM J. Optim.*, 27, 1459–1484.
- Yang, B., Zhao, X., Li, X., and Sun, D. F. (2025), “An accelerated proximal alternating direction method of multipliers for optimal decentralized control of uncertain systems,” *J. Optim. Theory Appl.*, 204, 1–37.
- Yang, L., Li, J., Sun, D. F., and Toh, K.-C. (2021), “A fast globally linearly convergent algorithm for the computation of Wasserstein barycenters,” *J. Mach. Learn. Res.*, 22, 1–37.

- Ye, J. and Li, J. (2014), “Scaling up discrete distribution clustering using ADMM,” in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 5267–5271, IEEE.
- Ye, J., Wu, P., Wang, J. Z., and Li, J. (2017), “Fast discrete distribution clustering using Wasserstein barycenter with sparse support,” *IEEE Trans. Signal Process.*, 65, 2317–2332.
- Zhu, M. and Chan, T. (2008), “An efficient primal-dual hybrid gradient algorithm for total variation image restoration,” *UCLA Cam Report*, 34.